Titre: Title:	Méthodes d'apprentissage appliquées aux heuristiques de recherche pour les problèmes de satisfaction de contraintes
Auteur: Author:	Ronan Le Bras
Date:	2009
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Le Bras, R. (2009). Méthodes d'apprentissage appliquées aux heuristiques de recherche pour les problèmes de satisfaction de contraintes [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/241/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/241/
Directeurs de recherche: Advisors:	Gilles Pesant
Programme: Program:	Génie informatique

UNIVERSITÉ DE MONTRÉAL

MÉTHODES D'APPRENTISSAGE APPLIQUÉES AUX HEURISTIQUES DE RECHERCHE POUR LES PROBLÈMES DE SATISFACTION DE CONTRAINTES

RONAN LE BRAS
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION DU DIPLÔME DE
MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2009

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

MÉTHODES D'APPRENTISSAGE APPLIQUÉES AUX HEURISTIQUES DE RECHERCHE POUR LES PROBLÈMES DE SATISFACTION DE CONTRAINTES

présenté par : LE BRAS Ronan
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury constitué de :

M. GAGNON Michel, Ph.D., président.
M. PESANT Gilles, Ph.D., membre et directeur de recherche.
M. PAL Christopher J., Ph.D., membre.

À ma femme, merveilleuse source d'inspiration
et relectrice intraitable...

Remerciements

J'aimerais exprimer ma gratitude envers ceux qui m'ont aidé à compléter ce mémoire.

En premier lieu, je suis sincèrement reconnaissant envers mon directeur de recherche, Professeur Gilles Pesant, notamment pour le partage de ses connaissances et son expertise, mais aussi pour la confiance qu'il m'a accordée. Ses précieux conseils m'ont guidé tant dans le cadre de ce mémoire que dans ma décision d'orientation académique. Tout simplement, je n'aurais pas pu réaliser les accomplissements tout au long de ce parcours sans sa contribution.

Par ailleurs, je souhaite remercier les professeurs Michel Gagnon et Christopher J. Pal d'avoir accepté d'être membres du jury.

J'adresse également un merci tout particulier à Alessandro Zanarini pour son importante contribution à ce projet, notamment par le biais de discussions constructives et de brassages d'idées très productifs. De plus, je désire exprimer ma reconnaissance envers lui pour avoir généreusement mis à ma disposition le code source nécessaire aux expérimentations.

Ensuite, je souhaite remercier les membres du laboratoire Quosseça, et tout particulièrement Marie-Claude Côté, Simon Boivin et Arnaud Malapert pour les conseils qu'ils m'ont donnés tout au long de ce parcours de recherche et pour l'avoir ponctué d'activités qui n'y étaient pas reliées.

Mes remerciements vont naturellement également au Conseil de Recherches en Sciences Naturelles et en Génie (CRSNG) du Canada pour m'avoir octroyé la bourse BESC-M Alexander-Graham-Bell pour la complétion de mon programme de maîtrise-recherche.

Je remercie également à la fois le programme FCI-Relève ainsi que le Centre Interuniversitaire de Recherche sur les Reseaux d'Entreprise, la Logistique et le Transport (CIRRELT) pour avoir financé les structures matérielles ayant permis à réaliser les expérimentations ménées dans le cadre de ce projet.

Enfin, je suis reconnaissant envers les participants du projet pilote "Thèses et mémoires électroniques" de l'École Polytechnique pour avoir mis à disposition le modèle de document que j'ai utilisé lors de la rédaction de ce mémoire.

Résumé

Motivation : La programmation par contraintes (PPC) propose un cadre formel pour représenter et résoudre des problèmes combinatoires concrets tels que la conception d'horaires de personnel d'hôpital ou l'allocation des portes d'embarquement dans un aéroport. Un problème de satisfaction de contraintes se modélise à l'aide de relations logiques, ou contraintes, entre des variables. La résolution du problème revient alors à identifier une solution qui respecte ces contraintes. Même si l'utilisation de techniques d'inférence puissantes, comme les algorithmes de filtrage, permettent de réduire sensiblement l'espace des solutions, cela reste insuffisant. Il convient alors de guider la recherche à l'aide d'heuristiques.

Enjeux et Contexte théorique : Cambazard et Jussien (2005) justifient l'intérêt porté aux heuristiques de recherche en les qualifiant de « Saint Graal » à la fois des communautés de recherche opérationnelle (RO) et de programmation par contraintes (PPC). Parmi les meilleures heuristiques actuelles figurent *Impact-Based Search* (IBS - Refalo (2004)), qui utilise la réduction de domaine après branchement, et `maxSD` (Zanarini et Pesant (2007)), qui utilise le dénombrement des solutions des contraintes. D'autres heuristiques se basent quant à elles sur une estimation de la distribution de solutions, comme les méthodes *Belief-Propagation* (BP - Kschischang *et al.* (2001)) et *Survey-Propagation* (SP - Mezard *et al.* (2002)). Ces méthodes, dites d'inférence, se sont notamment distinguées pour la résolution de problèmes de satisfaction booléenne. Récemment, une variante de ces deux méthodes, dénommée Expectation-Maximization Belief-Propagation, a été proposée par Hsu *et al.* (2007).

Enfin, ces méthodes d'inférence permettent également de déterminer certaines caractéristiques de la structure du problème que l'on cherche à résoudre. À titre d'exemple, de telles estimations permettent ainsi d'identifier les variables dites backbones (Kilby *et al.* (2005)), qui sont les variables qui prennent toujours la même valeur quelle que soit la solution. Ainsi, une estimation de la distribution des solutions ne fournit pas seulement de l'information utile pour une heuristique, mais également de l'information sur la structure sous-tendante du problème.

Problématique et Objectifs : Comment guider efficacement la recherche de solutions de problèmes combinatoires difficiles tout en s'affranchissant de la structure du problème ? Dans le but de répondre à cette problématique, l'objectif principal de ces travaux est de fournir des heuristiques de recherche efficaces et génériques qui guident la recherche vers des zones prometteuses, et permettent ainsi de faciliter la résolution de problèmes complexes pratiques. De façon plus détaillée, les objectifs spécifiques consistent en (1) l'élaboration de nouvelles méthodes d'inférence (notamment en agrégeant l'information des contraintes au niveau du problème global quelle que soit la structure du problème et les contraintes considérées), (2) le développement d'un framework de tests des heuristiques (de façon à estimer la distribution des solutions d'un ensemble d'instances de problèmes combinatoires), et (3) l'évaluation des heuristiques.

Contributions théoriques : En se basant sur le cadre théorique proposé par Hsu *et al.* (2007), nous développons plusieurs heuristiques de recherche qui améliorent significativement ces résultats antérieurs, et qui permettent d'étendre ce cadre théorique à n'importe quel problème de satisfaction de contraintes. Une des méthodes que nous proposons ici (EMBP-Gsup) constitue notamment l'heuristique de recherche la plus consistante sur l'ensemble des problèmes considérés, et ce, comparativement à ce qui constitue (au meilleur de nos connaissances) l'état de l'art en matière d'heuristiques pour ces mêmes problèmes. Ces contributions font d'ailleurs l'objet d'une publication (LeBras *et al.* (2009)).

Abstract

Motivation Constraint Programming (CP) is a paradigm to model and solve pratical combinatorial problems, such as nurse scheduling or airport gate assignment. CP models such problems as Constraint Satisfaction Problems (CSPs), i.e. a set of relations between variables. Solving a CSP then becomes equivalent to finding a solution that satisfies all relations. Although strong inference techniques such as filtering algorithms lead to a much smaller solution space, the solution space typically remains too large to be explored exhaustively. This is where search heuristics come in and guide the search toward promising areas.

Significance and Theoritical Background Cambazard and Jussien (2005) emphasize the importance of search heuristics when they refer to them as the "Holy Grail" of both Operations Research (OR) and Constraint Programming (CP) communities. Two of the best current search heuristics are *Impact-Based Search* (IBS - Refalo (2004)) which exploits domain reduction, and `maxSD` (Zanarini and Pesant (2007)), which exploits constraint-based solution counting. Other heuristics are designed to estimate the distribution of solutions, such as *Belief-Propagation* (BP - Kschischang *et al.* (2001)) and *Survey-Propagation* (SP - Mezard *et al.* (2002)). These inference methods have been particularly suitable to Satisfiability (SAT) problems. Recently, Hsu *et al.* proposed *Expectation-Maximization Belief-Propagation* (EMBP), a variation of these two methods.

In addition, these inference methods also provide information about the underlying structure of the problem. For example, such estimates enable to detect *backbone* variables (Kilby *et al.* (2005), i.e. that always take the same value regardless of the solution. As a result, these estimates not only direct the search but also provide structural information about the solution space.

Purposes How to efficiently guide the search in the solution space of hard combinatorial problems whatever their underlying structure? The first purpose of this work is to provide efficient and generic search heuristics to guide the search toward promising areas and to solve hard practical problems. More specifically, the goal is to (1) develop new inference methods (which combine information from any kind of con-

straints imposed by the problem), (2) implement a framework to test these heuristics (so that we compute these heuristics over a variety of instances), and (3) assess the quality of these heuristics.

Contributions Building upon the theoritical framework that has been proposed by Hsu *et al.* (2007), we derive new search heuristics that significantly improve the previous results obtained with this framework, and extend it to any CSP. One of the methods we propose (EMBP-Gsup) turns out to provide the most consistent results for the problems we consider when compared to what is (to the best of our knowledge) the current state of the art for solving these problems. This contribution has been published in (LeBras *et al.* (2009)).

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et abréviations

BP	Belief-Propagation
CP	Constraint Programming
CSP	Constraint Satisfaction Problem
DAG	Directed Acyclic Graph
DFA	Deterministic Finite Automaton
EM	Expectation-Maximization
EMBP	Expectation-Maximization Belief-Propagation
IBS	Impact-Based Search
MaxSD	Maximum Solution Density
MCMC	Markov Chain Monte-Carlo
OR	Operations Research
PPC	Programmation Par Contraintes
QWH	Quasi-group With Holes
RSC-LA	Restricted Singleton Consistency Look-Ahead
SP	Survey-Propagation

Chapitre 1

INTRODUCTION

1.1 Motivation

La programmation par contraintes (PPC) propose un cadre formel pour représenter et résoudre des problèmes combinatoires concrets. La conception d'horaires de personnel d'hôpital, de rotation d'équipage aérien, d'ordonnancement de production, d'allocation de fréquences pour les cellulaires ou encore d'allocation des portes d'embarquement dans un aéroport sont autant de problèmes qui témoignent de la multitude d'applications de ce domaine de recherche. Ce sont autant d'applications qui motivent ces travaux, sachant que ceux-ci peuvent se traduire concrêtement par des impacts socio-économiques importants.
Un problème de satisfaction de contraintes se modélise à l'aide de relations logiques, ou contraintes, entre des variables. La résolution du problème revient alors à identifier une solution qui respecte ces contraintes. En théorie donc, la programmation par contraintes permet de s'affranchir de la méthode de résolution. Ultimement, il suffit d'énoncer le problème à la machine pour que celle-ci trouve une solution. En pratique, le fléau de la dimension entre en jeu : l'espace des possibilités grandissant exponentiellement avec le nombre de variables, il s'avère impraticable d'énumérer toutes les possibilités. Même si l'utilisation de techniques d'inférence puissantes, comme les algorithmes de filtrage, permettent de réduire sensiblement l'espace des solutions, cela reste insuffisant. Il convient alors de guider la recherche à l'aide d'heuristiques.

1.2 Enjeux

Le *Handbook of Contraint Programming* (Rossi *et al.* (2006)) décrit les qualités reconnues d'une bonne heuristique de recherche, telles que le « *first-fail principle* » (en d'autres termes, « essayer d'abord là où l'on est le plus susceptible d'échouer », ou encore « commencer par ce qui semble le plus difficile à satisfaire »). Ce livre

présente les principales heuristiques génériques, i.e. qui sont dites indépendantes de l'application. Cambazard et Jussien (2005) qualifient les heuristiques de recherche génériques comme étant le « Saint Graal » à la fois des communautés de recherche opérationnelle (RO) et de programmation par contraintes (PPC). Ils justifient ainsi tout l'intérêt porté aux méthodes guidant la recherche dans un espace de solutions. En 2004, Refalo introduit une heuristique générique dénommée *Impact-Based Search* (IBS) qui se révèlera être l'état de l'art pour la résolution de plusieurs problèmes combinatoires. Plus récemment, une comparaison de l'efficacité de différentes heuristiques a été proposée par Zanarini et Pesant (2007).

Par ailleurs, certaines heuristiques présentes dans la littérature se basent sur une estimation de la distribution de solutions dans l'optique de prendre une décision de branchement qui mène à une zone prometteuse. Dans cet environnement probabiliste, on retrouve les méthodes d'inférence dites de « *message-passing* », qui permettent d'estimer les probabilités marginales des variables du problème (i.e. la fonction de distribution d'une variable dans l'espace des solutions). Par exemple, la méthode *Belief-Propagation* (Pearl (1988), Kschischang *et al.* (2001)) a été à l'origine développée afin de résoudre les problèmes d'inférence que l'on retrouve par exemple dans la théorie de correction d'erreur d'encodage ou en vision par ordinateur. Kask *et al.* (2004) ont montré par la suite que cette méthode est particulièrement efficace lorsqu'appliquée aux problèmes combinatoires et utilisée en tant qu'heuristique de recherche. Mezard *et al.* (2002) ont inventé une méthode similaire appelée *Survey Propagation*, qui est actuellement la référence pour la résolution de problèmes de satisfaction booléenne (Braunstein *et al.* (2005), Kroc *et al.* (2007)). Enfin, plus récemment, une variante de ces deux méthodes, dénommée *Expectation-Maximization Belief-Propagation*, a été proposée par Hsu *et al.*, qui ont tout d'abord appliqué cette méthode à des problèmes SAT (Hsu et Mcilraith (2006)), puis qui l'ont étendue à des problèmes de satisfaction de contraintes (Hsu *et al.* (2007)).

L'information fournie par les méthodes d'inférence citées précédemment permet d'inférer certaines caractéristiques de la structure du problème que l'on cherche à résoudre. En effet, si l'on est capable d'estimer de façon fiable la distribution des solutions d'un problème, on connaît alors la probabilité qu'une variable prenne une certaine valeur dans une solution aléatoire du problème. À titre d'exemple, de telles

estimations permettent ainsi d'identifier les variables qui prennent toujours la même valeur quelle que soit la solution (i.e. qui ont une probabilité de 1 de prendre cette valeur). Ainsi, une estimation de la distribution des solutions ne fournit pas seulement de l'information utile pour une heuristique, mais également de l'information sur la structure sous-jacente du problème.

Ceci souligne le potentiel intéressant de l'estimation de la distribution des solutions et constitue le principal enjeu de ces travaux de recherche.

1.3 Problématique et objectifs de recherche

Dans le cadre de nos travaux de recherche, nous nous intéressons à la manière de guider efficacement la recherche de solutions de problèmes combinatoires difficiles tout en nous affranchissant de la structure du problème. Pour fins d'évaluation, les deux principaux critères imposés sont l'efficacité et le caractère générique de l'heuristique. Combiner ces deux qualités constitue l'une des principales difficultés. En effet, une heuristique spécifique à une contrainte donnée peut s'avérer particulièrement efficace, sans pouvoir être étendue (ou difficilement) à d'autres contraintes. À l'inverse, une heuristique générique s'applique à n'importe quel problème mais généralement aux dépens de son efficacité.

Afin de répondre à cette problématique, l'objectif principal de nos travaux est de fournir des heuristiques de recherche efficaces et génériques qui guident la recherche vers des zones prometteuses, et faciliteront ainsi la résolution de problèmes complexes pratiques (tels que ceux mentionnés à la section 1.1).

Plus précisément, les objectifs spécifiques liés à nos travaux sont les suivants. Tout d'abord, nous souhaitons élaborer de nouvelles méthodes d'inférence qui agrègent l'information des contraintes au niveau du problème global (et non pas localement, i.e. au niveau d'une contrainte par exemple) et cela quelle que soit la structure du problème et les contraintes considérées. Ensuite, nous désirons estimer la distribution des solutions d'un ensemble de problèmes combinatoires à l'aide de ces méthodes. Il s'agira enfin de comparer les estimations obtenues précédemment aux valeurs exactes et de les évaluer dans le contexte d'une heuristique de recherche.

1.4 Plan du mémoire

Le présent mémoire est organisé comme suit. La première partie traite du contexte théorique dans lequel se situent ces travaux. Ensuite, la seconde partie présente les méthodes proposées tandis que la troisième partie développe les résultats expérimentaux de ces méthodes. Finalement, les conclusions, discussions et pistes de recherche futures font l'objet de la dernière partie.

Chapitre 2

CONTEXTE THÉORIQUE

Ce chapitre introduit à la section 2.1 le paradigme de la programmation par contraintes. Ensuite, la section 2.2 présente l'intérêt de l'estimation de la distribution des solutions d'un CSP, tandis que la section 2.3 est dédiée aux méthodes d'inférence, ayant pour but d'estimer cet espace des solutions.

2.1 Programmation par contraintes

2.1.1 Principe

La programmation par contraintes (PPC) représente un paradigme de résolution de problèmes combinatoires. L'idée centrale de ce paradigme est de définir l'ensemble des contraintes qu'une solution acceptable doit satisfaire. Cette approche permet ainsi de s'affranchir de la méthode de résolution en laissant le soin au programme de trouver une solution satisfaisant ces contraintes. En ce sens, le langage utilisé en PPC est dit déclaratif, c'est-à-dire que l'on ne précise pas comment effectuer l'opération mais plutôt quelles contraintes respecter. « La programmation par contraintes représente une des avancées scientifiques qui se rapproche le plus du Saint Graal de la programmation : l'utilisateur définit le problème, l'ordinateur le résout » (Freuder (1996)). Cette citation de Eugène Freuder, précurseur de la PPC, rappelle le principe de la PPC et souligne tout le potentiel de ce paradigme de résolution de problèmes.

2.1.2 Modélisation

La programmation par contraintes consiste à définir un problème en énonçant les relations logiques (que l'on appelle contraintes) qui lient plusieurs variables. Ainsi, un des avantages de ce paradigme réside dans le fait qu'une contrainte représente généralement une composante naturelle de problèmes combinatoires. De façon plus formelle, un problème de satisfaction de contraintes (ou *Constraint Satisfaction Prob-*

lem - CSP) se modélise à l'aide de trois principales composantes (cf. Définition 1) : un ensemble de contraintes qui régissent le problème, un ensemble de variables intervenant dans ces contraintes ainsi que les domaines respectifs de ces variables.

Définition 1. *Un problème de satisfaction de contraintes (CSP) consiste en :*

- *un ensemble fini de variables*[1] $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
- *définies sur des domaines* $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ *de telle façon que* $x_i \in D_i$ *pour tout* i,
- *ainsi qu'un ensemble fini de contraintes* $\mathcal{C}$, *chacune définie sur un sous-ensemble de* $\mathcal{X}$.

Selon cette notation, on notera par le triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ un tel CSP.

Définition 2. *Une contrainte* $C_i \in \mathcal{C}$ *est un sous-ensemble* $T(C_i)$ *du produit Cartésien des domaines des variables qui interviennent dans* C_i. *On note* $X(C_i)$ *l'ensemble des variables intervenant dans* C_i *et on appelle* tuple $\tau \in T(C_i)$ *une combinaison valide de valeurs pour les variables dans* $X(C_i)$.

Par exemple, un carré latin consiste en une grille $n \times n$ telle que les valeurs de 1 à n apparaissent une et une seule fois sur chaque ligne et sur chaque colonne. On peut modéliser le problème du carré latin de la façon suivante. Pour chacune des n^2 cellules de la grille, on définit une variable prenant une valeur entre 1 et n ; ce qui fixe ainsi à la fois $\mathcal{X}$ et $\mathcal{D}$. On exprime ensuite les relations entre ces variables (i.e. $\mathcal{C}$). Pour chaque ligne (colonne) de la grille, une combinaison valide de valeurs des variables de cette ligne (colonne) est telle que ces valeurs sont différentes. De façon formelle, cette formulation correspond à :

$$
\begin{array}{ll}
\mathcal{X} = \{x_{11}, x_{12} \ldots x_{nn}\} & \\
D_{ij} = \{1, 2, \ldots, n\} & \forall 1 \leq i, j \leq n \\
\texttt{alldifferent}((x_{ij})_{1 \leq j \leq n}) & \forall 1 \leq i \leq n \\
\texttt{alldifferent}((x_{ij})_{1 \leq i \leq n}) & \forall 1 \leq j \leq n \\
x_{ij} = d & \forall (i, j, d) \in S
\end{array}
$$

où S répresente l'ensemble des cellules préfixées.

[1] Par abus de notation, on ne distingue pas une variable de la valeur qui lui est assignée.

Dans le but d'alléger la notation, les variables sont ici indexées selon la ligne et la colonne auxquelles elles appartiennent. Cependant, les définitions précédentes utilisent une indexation simple de ces variables, de 1 à $|\mathcal{X}|$. Par la suite, nous utilisons au besoin une de ces deux indexations, puisque cela n'introduit aucune ambiguité.

Cette modélisation fait appel à la contrainte globale (i.e. définie sur un nombre quelconque de variables) `alldifferent`. Cette contrainte assure que les valeurs prises par chacune des variables qui interviennent dans la contrainte sont toutes différentes. De façon plus formelle, et en reprenant la notation de la définition 2, cette contrainte peut être définie comme suit :
$T(\texttt{alldifferent}(x_1, x_2, \ldots, x_n)) = \{(x_1, x_2, \ldots, x_n) | \forall 1 \leq i \leq n, \forall i < j \leq n, x_i \neq x_j\}$

2.1.3 Résolution

Un autre avantage de la programmation par contraintes est que, une fois le problème modélisé, un vaste choix de techniques de résolution est disponible.

La phase de résolution est marquée par deux phases itératives : la propagation et la recherche. La propagation consiste à inférer des simplifications du problème étudié à un moment donné de la recherche par retour arrière (généralement par le biais de réductions de domaines) tandis que la recherche revient à prendre une décision de simplification en différents sous-problèmes.

a) Propagation

La phase de propagation est effectuée à l'aide d'algorithmes de filtrage successivement appliqués aux contraintes (généralement globales) du problème. Ces algorithmes permettent (en général) de supprimer des valeurs incohérentes (i.e. qui n'appartiennent à aucune solution) des domaines des variables, ce qui aura potentiellement pour effet de provoquer d'autres réductions de domaines.

Plusieurs niveaux de cohérence peuvent alors être définis, afin de déterminer les valeurs incohérentes. Nous définissons ici quatre niveaux de cohérence auxquels nous ferons référence dans la suite de ces travaux.

Cohérence d'arc Tout d'abord, la cohérence d'arc est un niveau de cohérence élémentaire et est définie spécifiquement sur une contrainte binaire $c \in \mathcal{C}$, i.e. $\mathcal{X}(c) =$

$\{x_i, x_j\}$. La cohérence d'arc assure que toute valeur du domaine d'une des deux variables est compatible (d'après la contrainte c) avec au moins une valeur du domaine de l'autre variable. On a la cohérence d'arc sur c si et seulement si :
$D_i \subseteq \{v \in D_i | (\{v\} \times D_j) \bigcap c \neq \emptyset\}$ et réciproquement pour D_j.

Si ce n'est pas le cas, on supprime la ou les valeurs de D_i et D_j qui sont responsables de cet échec.

Cohérence de domaine La cohérence de domaine généralise la notion de cohérence d'arc à une contrainte d'arité quelconque. D'ailleurs, la cohérence de domaine est également appelée *cohérence d'arc généralisée* dans la littérature. De la même façon, on a la cohérence de domaine sur c d'arité k si et seulement si :
$\forall 1 \leq i \leq k, D_i \subseteq \{v \in D_i | D_1 \times \ldots \times D_{i-1} \times \{v\} \times D_{i+1} \times \ldots \times D_k \bigcap c \neq \emptyset\}$.

Cohérence de bornes La cohérence de bornes fait appel à la définition de l'ensemble $D_{\mathcal{S}}$, défini pour un ensemble $\mathcal{S}$ et qui correspond à un ensemble relaxé selon $\mathcal{S}$ du domaine d'une variable. On pose $D_{\mathcal{S}}(x) = \{v \in \mathcal{S} | min(D(x)) \leq v \leq max(D(x))\}$. La cohérence de bornes ($\mathcal{S}$) assure que, pour une contrainte quelconque c d'arité k, les valeurs extrêmes (minimale et maximale) du domaine d'une variable sont cohérentes avec les domaines $D_{\mathcal{S}}$ des $k-1$ autres variables. On établit la cohérence de bornes (D) sur c d'arité k de la façon suivante :
$\forall 1 \leq i \leq k, \{min(D_i), max(D_i)\} \subseteq \{v \in D_i | D_{\mathcal{S}}(x_1) \times \ldots \times D_{\mathcal{S}}(x_{i-1}) \times \{v\} \times D_{\mathcal{S}}(x_{i+1}) \times \ldots \times D_{\mathcal{S}}(x_k) \bigcap c \neq \emptyset\}$.

Par exemple, la cohérence de bornes ($\mathbb{R}$) est celle que nous utilisons pour les contraintes `knapsack` du problème du Carré Magique (cf. la section des résultats expérimentaux). Des résultats sur ce même problème et utilisant ce niveau de cohérence sont présentés dans Pesant et Quimper (2008). Ainsi, on obtient $D_{\mathbb{R}}(x) = \{v \in \mathbb{R} | min(D(x)) \leq v \leq max(D(x))\}$ pour le domaine relaxé des variables et la cohérence de bornes ($\mathbb{R}$) assurent que :
$\forall 1 \leq i \leq k, \{min(D_i), max(D_i)\} \subseteq \{v \in D_i | D_{\mathbb{R}}(x_1) \times \ldots \times D_{\mathbb{R}}(x_{i-1}) \times \{v\} \times D_{\mathbb{R}}(x_{i+1}) \times \ldots \times D_{\mathbb{R}}(x_k) \bigcap c \neq \emptyset\}$.

Cohérence X singleton Enfin, une dernière définition de niveau de cohérence présentée ici est la cohérence X singleton. On dit que le CSP $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ est cohérent

selon cette cohérence si et seulement si on a la cohérence X (où X représente un niveau de cohérence en particulier) pour chacun des sous-CSPs $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \bigcup \{x_i = v\} \rangle$ pour tout x_i de $\mathcal{X}$ et pour tout v de D_i.

b) Recherche par retour arrière

La propagation à elle-seule ne suffit pas à résoudre des problèmes complexes.

En effet, suite à la phase de propagation, chaque élément du produit cartésien des domaines des variables ne constitue pas forcément une solution valide. Il n'est pas certain que l'on puisse étendre une solution en choisissant d'assigner arbitrairement une valeur du domaine d'une variable et de propager cette décision. En particulier, une phase de propagation peut mener à un domaine d'une variable qui est réduit à l'ensemble vide, ce qui produit alors un échec et force à reconsidérer les décisions précédentes.

On procède alors à un retour-arrière (ou *backtracking*), qui permet d'explorer l'arbre de recherche. Chaque nœud de l'arbre est appelé nœud de branchement, puisqu'il divise le problème courant en plusieurs alternatives disjointes. Une feuille de l'arbre de recherche correspond à une affection de l'ensemble des variables. Ainsi, une feuille constitue une solution du CSP si et seulement si cette affectation totale respecte toutes les contraintes.

Types de branchement : Théoriquement, à chaque nœud de branchement, une décision de branchement représente une simplification de l'espace en au moins deux sous-problèmes. En effet, une telle décision revient à créer une partition de l'espace de recherche, et chacun des ensembles disjoints de cette partition représente une branche associée à ce nœud de branchement. Les principaux types de branchement présents dans la littérature sont directement reliés à l'utilisation de domaines des variables comme relaxation du problème à chaque nœud de l'arbre. Ainsi, les types de branchement les plus courants définissent une réduction du domaine d'une variable. Cela revient ainsi à ajouter une contrainte unaire, i.e. ne faisant intervenir qu'une seule variable. Par exemple, si on a $x_1 \in \{1, 2\}$ et $x_2 \in \{1, 2\}$, la contrainte $x_1 \leq 1$ représente une réduction du domaine de la variable x_1, par opposition à $x_1 \neq x_2$ qui restricte la paire (x_1, x_2) aux tuples $\{(1, 2), (2, 1)\}$ mais ne réduit pas les domaines

des variables. L'exemple 2.1 ci-dessous illustre les trois principaux types de branchement utilisés, pour lesquels la variable x_4 fait l'objet de la décision de branchement. La figure a) présente le branchement énumératif, où chacune des branches représente une des valeurs de D_4. Le brachement binaire (figure b)) définit deux branches complémentaires : la première correspond à l'assignation d'une valeur de D_4 à x_4, et la seconde est la négation de la première. Enfin, la figure c) illustre le branchement par fragmentation de domaines qui divise typiquement le domaine de la variable x_4 en deux sous-ensembles de taille arbitraire. Dans le cadre de ce travail, nous nous intéressons uniquement à un branchement de type binaire (sur le domaine d'une variable) (Fig 2.1b).

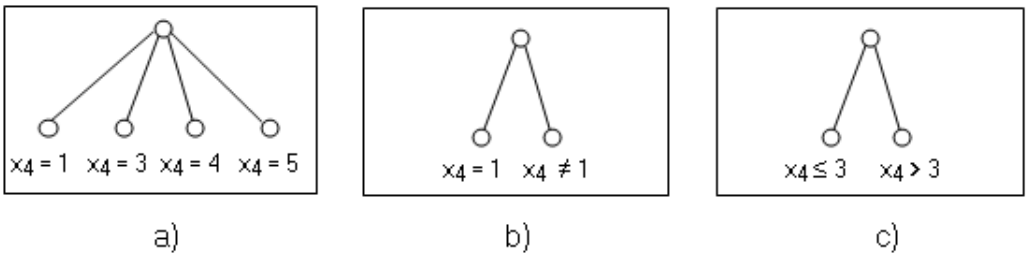


FIGURE 2.1 Principaux types de branchement appliqués à la variable x_4, pour laquelle $D_4 = \{1, 3, 4, 5\}$: a) Énumératif, b) Binaire et c) Par fragmentation de domaine

Heuristiques de recherche : Les heuristiques de recherche traditionnelles, utilisées dans le cadre d'un branchement binaire, ont pout rôle d'ordonner les variables ainsi que les valeurs à leur assigner.

Les principes de base actuellement reconnus qui régissent la sélection du couple variable-valeur sont les suivants :

Choix de la variable :	
« First-fail principle »	O n choisit la variable qui semble la plus difficile à satisfaire, notamment en espérant obtenir un arbre de recherche le plus petit possible. De cette manière, on désire s'occuper de la partie difficile à satisfaire en premier.
Choix de la valeur :	
« Succeed-first principle »	On choisit la valeur qui semble la plus prometteuse pour trouver une solution. De cette façon, on espère explorer le moins de nœuds de l'arbre engendré précédemment par le choix de l'ordre des variables.

Parmi les principales heuristiques illustrant le premier principe, on retrouve dom, dom/deg et dom/wdeg. L'heuristique dom (Golomb et Baumert (1965)) sélectionne la variable ayant le domaine courant le plus petit. Introduisant la notion de degré d'une variable comme étant le nombre de contraintes impliquant cette variable ainsi qu'une autre dont le domaine n'est pas réduit à un singleton, l'heuristique dom/deg (Bessière et Régin (1996)) classe les variables selon le ratio de la taille du domaine sur le degré. Enfin, la notion de degré pondéré introduit un poids à chaque contrainte correspondant au nombre de fois qu'une contrainte est responsable d'un échec. Cette notion permet de définir l'heuristique dom/wdeg (Boussemart *et al.* (2004) comme étant le ratio entre la taille du domaine d'une variable et le degré pondéré de cette variable.

Parmi les principales heuristiques illustrant le deuxième principe, on retrouve notamment maxSD, qui correspond à choisir le couple variable-valeur selon le nombre de solutions auxquelles ce couple appartient pour une contrainte en particulier. Pour le choix de la valeur à assigner à une variable en particulier, Ginsberg *et al.* (1990) proposent quant à eux de prendre la valeur v qui maximise le produit des tailles des domaines restants des autres variables après l'affectation $x = v$. Cette notion fait appel à la définition suivante :

Définition 3. *À un noeud de branchement donné, on note $\tilde{D}_{x_i=v}(x_j)$ l'ensemble des valeurs restantes du domaine de la variable x_j après propagation de la contrainte $x_i = v$.*

L'heuristique de choix de valeur proposée par Ginsberg *et al.* (1990) est parfois appelée promise, et correspond par conséquent à :

$$\arg\max_{v} \prod_{j \neq i} card(\tilde{D}_{x_i=v}(x_j))$$

Similairement, Frost et Dechter (1995) suggèrent une autre heuristique illustrant le deuxième principe et qui revient à remplacer le produit précédent par une somme. Cette heuristique, que l'on dénote ici min-conflict, correspond à :

$$\arg\max_{v} \sum_{j \neq i} card(\tilde{D}_{x_i=v}(x_j))$$

Bien que les heuristiques énoncées ici figurent parmi celles qui sont le plus présentes dans la littérature, cette liste est loin d'être exhaustive. La section suivante aborde le sujet des critères d'évaluation de telles heuristiques, qui ont notamment joué le rôle de la « sélection naturelle » des meilleures heuristiques de recherche.

Évaluation de la qualité d'une heuristique Théoriquement et tel que défini dans le *Handbook of Constraint Programming* (Rossi *et al.* (2006)), on qualifie d'optimal l'ordre des variables ou des valeurs qui permet de parcourir le moins de nœuds de branchement (parmi tous les ordres possibles) lorsque l'on cherche une solution ou que l'on cherche à prouver qu'il n'existe aucune solution.
Cependant, il est également important de considérer le travail réalisé à chaque nœud et de privilégier une heuristique moins coûteuse à chaque décision de branchement. En effet, une heuristique pourrait par exemple trouver une solution par force brute et ensuite instancier les variables une à une telles qu'elles apparaissent dans la solution trouvée. Notamment, il est à noter que déterminer un ordre optimal sur les variables ou les valeurs est au moins aussi difficile que de résoudre le problème, tout comme de déterminer la prochaine variable à instancier de façon optimale (Rossi *et al.* (2006)).
Enfin, étant donné que l'on ne dispose pas d'un temps illimité, il est nécessaire de fixer un temps maximal lors d'une exécution. C'est pourquoi il se peut que ce temps alloué soit insuffisant à une heuristique pour trouver une solution ou prouver qu'il n'en existe aucune.
Ainsi, dans le cadre de l'évaluation de la qualité d'une heuristique, nous avons retenu le nombre de retours arrières (ou backtrack), le temps d'exécution ainsi que le nombre

d'instances résolues.
L'extensibilité (ou `scalability`) est un autre élément intéressant dans l'évaluation d'une heuristique, bien que moins quantifiable et qui peut transparaître dans le nombre de retours arrières. Une façon indirecte d'évaluer ce facteur est de considérer des problèmes dont les instances sont de taille intéressante, où les heuristiques de recherche dites informées prennent largement le dessus sur les heuristiques de recherche non informées. Par exemple, tandis que les précédents résultats résolvent des instances de carrés latins de taille 15×15 de Hsu *et al.* (2007), nous avons réalisé nos expérimentations sur des exemplaires de taille 30×30.

Dans cette section, nous avons présenté les principaux principes et mécanismes de la PPC et introduit la notion d'heuristique de recherche dans le cadre de la résolution d'un CSP. La section suivante traite des estimations de l'ensemble des solutions d'un CSP et aborde le potentiel de ces estimations en termes d'heuristique de recherche.

2.2 Estimation de la distribution des solutions

Cette section définit tout d'abord les distributions (ou marginales) des variables d'un CSP, pour ensuite détailler l'utilisation potentielle de l'estimation de ces marginales et définir enfin comment évaluer la qualité d'une telle estimation.

2.2.1 Marginalisations exactes

Soit S l'ensemble des solutions d'un problème de satisfaction de contraintes (CSP). Si S est non vide, alors on peut définir la probabilité $P(x_i = v)$ qu'une variable x_i prenne la valeur v dans une solution aléatoire de S. On appellera *densité* cette probabilité.

Définition 4. *Soit S l'ensemble des solutions d'un CSP. Pour une solutions s de S, on note $s(x_i)$ la valeur prise par la variable x_i dans la solution s. Si $S \neq \emptyset$, la* densité *du couple variable-valeur $x_i = v$ correspond à :*
$P(x_i = v) = \frac{card(\{s \in S | s(x_i) = v\})}{card(S)}$

Ainsi, si le CSP admet au moins une solution, $P(x_i)$ définit une distribution de probabilité sur le domaine D_i. La probabilité $P(x_i = v)$ correspond à la densité des

solutions du problème complet qui vérifient $x_i = v$. Il est à noter que maxSD étend notamment cette notion de densité à une contrainte en particulier, comme étant la proportion de solutions d'une contrainte telles que x_i prend la valeur v.

Revenons sur le problème de carré latin, précédemment introduit à la section 2.1.2, et considérons l'exemple 1 présentéeci-dessous.

Exemple 1. *Carré latin de taille* 5, *avec* $S = \{(x_1, 4), (x_5, 2), (x_9, 2), (x_{12}, 2), (x_{16}, 2), (x_{17}, 1), (x_{18}, 3), (x_{19}, 4), (x_{20}, 5), (x_{22}, 4), (x_{23}, 2), (x_{24}, 5)\}$.

La figure 2.2 présente la version graphique de cette instance de carré latin

4				**2**
			2	
	2	x_{13}		
2	**1**	**3**	**4**	**5**
	4	**2**	**5**	

FIGURE 2.2 Exemple d'instance du problème de carré latin

Les solutions de cette instance sont données à la figure 2.3.

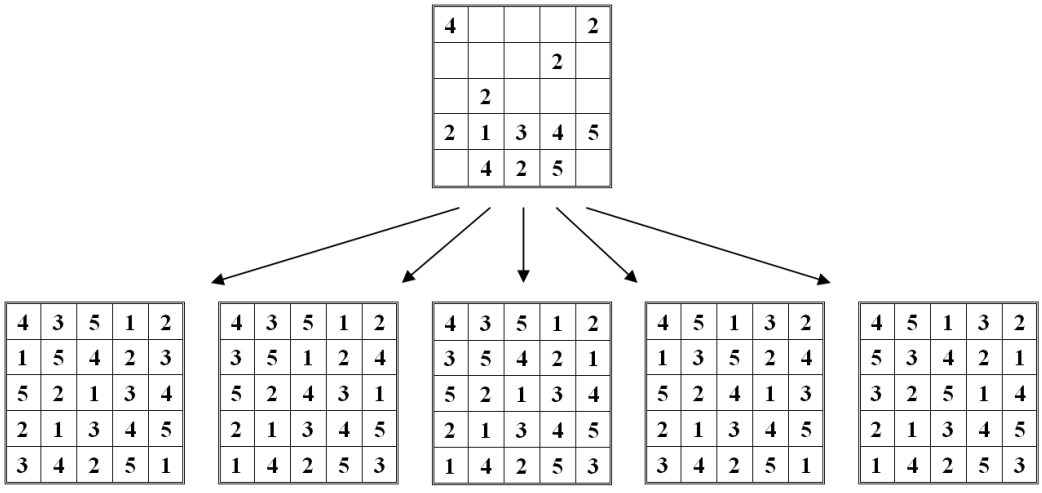


FIGURE 2.3 Solutions de l'exemple d'instance du problème de carré latin

D'après cet ensemble de solutions, la distribution de probabilité $P(x_{13})$ peut être déterminée de façon exacte et correspond à :

- $P(x_{13} = 1) = \frac{2}{5}$
- $P(x_{13} = 4) = \frac{2}{5}$
- $P(x_{13} = 5) = \frac{1}{5}$

Cependant, il est alors absurde de déterminer l'ensemble des solutions, pour calculer ces distributions de probabilités que l'on souhaite utiliser dans le cadre d'une heuristique de recherche d'une solution.

Similairement, supposons que l'on connaisse les distributions $P(x_i)$ de façon exacte à tout nœud de branchement. Alors, on peut par exemple décider de brancher sur n'importe quel couple variable-valeur ayant une probabilité non nulle (il en existe un étant donné que le problème possède au moins une solution). Le sous-problème obtenu de cette façon peut lui-même être étendu à une solution, puisque que le couple variable-valeur choisi apparaît dans au moins une solution du problème. C'est pourquoi, en répétant cette étape jusqu'à ce que toutes les variables soient assignées, on obtient alors une solution valide. Ainsi, déterminer la distribution des solutions d'un problème est au moins aussi difficile que de résoudre le problème lui-même.

De la même façon, à partir des distributions exactes, on est capable de détecter les *backbone variables* (Parkes (1997), Monasson *et al.* (1999), Kilby *et al.* (2005)), c'est-à-dire les variables qui ont toujours la même valeur quelle que soit la solution considérée du problème. Littéralement, le terme *backbone* se traduit par « colonne vertébrale », et réfère ici à la structure commune à toutes les solutions. Étant donné que même une approximation du caractère *backbone* d'une variable est un problème difficile en soit, le problème d'estimer la distribution des solutions est lui-même un problème difficile.

2.2.2 Utilisation de marginalisations approximées dans le cadre d'heuristiques

Déterminer les marginales exactes de l'ensemble des solutions est de façon générale intractable. C'est pourquoi l'objectif est ici non pas de les calculer de façon exacte,

mais bien de les approximer.

Définition 5. *On note $\theta_x(v)$ et on appelle* biais *l'estimation approximée de la densité $P(x = v)$.*

Définition 6. *On note θ_x la distribution de probabilité correspondant à l'ensemble des biais $\theta_x(v)$ pour chacune des valeurs possibles de x.*

Définition 7. *On note Θ et on appelle* étude *l'ensemble des distributions de biais θ_x, calculés pour chacune des variables x de $\mathcal{X}$.*

Une fois que l'on a obtenu une approximation de la distribution des solutions, i.e. après avoir complété une *étude*, il nous incombe de définir la façon d'utiliser cette information. Notre objectif est d'intégrer cette information de façon à guider la recherche. Tel que présenté dans la section précédente, deux décisions conceptuelles interviennent notamment ici : le choix de la stratégie de branchement et celui de l'ordre des variables et des valeurs.

Type de branchement utilisé En termes de type de branchement, nous avons adopté une stratégie binaire. La flexibilité quant au choix de la prochaine variable suite à un échec de la stratégie binaire par rapport à la stratégie d'énumération en fait une stratégie de branchement plus adaptée à l'évaluation d'une heuristique de recherche. En ce qui concerne la stratégie de fragmentation de domaine, celle-ci pourrait s'avérer être une piste intéressante. Cette stratégie nécessite cependant un paramètre supplémentaire (correspondant par exemple à la taille du sous-domaine sur lequel on branche) et face à une absence de dominance apparente d'une stratégie sur l'autre, nous avons privilégié la simplicité de la stratégie binaire.

Sélection du couple variable-valeur Les décisions relatives au classement des variables et des valeurs sont elles plus délicates et les possibilités sont plus larges. Admettons que l'on dispose d'une approximation des marginales. Ainsi, nous avons une estimation de la probabilité que la variable x prenne la valeur v dans une solution aléatoire du problème, et ce pour chacune des variables et pour chacune de leurs valeurs possibles respectives.

- `maxBias` : Une première possibilité, celle qui semble la plus intuitive, est de brancher sur le couple variable-valeur qui possède la probabilité la plus élevée. Nous appellerons cette stratégie de branchement `maxBias`.
- `maxStrength` : Une autre possibilité est de considérer le couple variable-valeur qui possède la force la plus élevée. On définit la force de la probabilité d'un couple variable comme la différence entre cette probabilité et l'équiprobabilité des valeurs du domaine d'une variable. Dans le cas de l'exemple de la figure 2.3, la force de la probabilité du couple $x_{13} = 1$ est $\frac{2}{5} - \frac{1}{3} = \frac{1}{15}$.
- `maxRegret` : Enfin, une troisième possibilité consiste à considérer la variable qui possède le plus grand regret. On définit le regret d'une variable comme étant la différence entre les deux probabilités les plus élevées de sa distribution. Dans l'exemple précédent, le regret de la variable x_{13} est égal à $\frac{2}{5} - \frac{2}{5} = 0$. Cette stratégie consiste alors à affecter à la variable de plus grand regret la valeur qui a la probabilité maximale. Le regret correspond en fait au coût (en termes d'objectif, qui correspond ici à maximiser la probabilité que le couple choisi appartienne à une solution) de choisir la deuxième meilleure valeur au lieu de la meilleure valeur.

2.3 Méthodes d'inférence

2.3.1 Raisonnement dans un environnement incertain

Un CSP permet de définir si une instanciation complète est valide (i.e. ne viole aucune contrainte) ou non. Ainsi, en reprenant la notation de la section précédente, un CSP définit une loi de probabilité jointe sur l'ensemble des variables du problème, i.e. $P(x_1, x_2, \ldots, x_n)$.

À chaque tuple $\tau = (x_1 = \tau_1, x_2 = \tau_2, \ldots, x_n = \tau_n) \in T(\mathcal{X})$, on associe la probabilité 1 si le tuple est un tuple valide et 0 sinon. C'est pourquoi estimer la distribution des solutions revient à déterminer les fonctions marginales de chacune des variables, i.e. $P(x_i), \forall x_i$.

Par sommation partielle, on peut donc calculer ces marginales de façon exacte. Ainsi, la distribution de probabilités conjointe complète permet de déduire une in-

férence exacte. Cependant, cette façon de faire devient rapidement impraticable dû à l'explosion combinatoire avec le nombre de variables.

Un CSP définit une loi jointe sur un ensemble de variables, et peut donc être modélisé indifféremment à l'aide d'un des trois plus courants modèles graphiques (ou *graphical models*) : un graphe de facteurs (ou *Factor Graph*), un champ aléatoire de Markov (ou *Pairwise Markov Random Field*) ou encore un réseau bayésien (ou *Bayesian Network*). Un graphe de facteurs est un graphe biparti qui représente la factorisation d'une fonction de plusieurs variables : chaque facteur représente un noeud connecté aux variables qui participent dans ce facteur. Un champ aléatoire de Markov est représenté à l'aide d'un graphe non-orienté reliant un ensemble de variables qui vérifient la propriété de Markov, notamment en termes d'indépendance conditionnelle. Enfin, un réseau bayésien est un graphe orienté dans lequel chaque arc répresente une relation causale entre deux variables. Utilisant généralement un de ces modèles graphiques pour exprimer une probabilité jointe, les méthodes d'inférence, et plus particulièrement les algorithmes par passage de messages (ou *message-passing algorithms*) ont pour but d'estimer les lois marginales pour chacune des variables. À l'aide de la mise à jour successive des probabilités associées à chacun des noeuds du modèle graphique, ces méthodes utilisent des messages pour propager localement l'information de chacun des noeuds, réalisant ainsi l'inférence souhaitée.

L'exemple suivant 2 illustre ces trois représentations graphiques pour un CSP constitué de cinq variables et deux contraintes. Par la suite, et par souci de concision et de clarté, on ne considère que le modèle (bien connu) des réseaux bayésiens.

Exemple 2. $\mathcal{X} = \{x_1, x_2, x_3, x_4, x_5\}$, $\mathcal{C} = \{C_1, C_2\}$, *avec* $X(C_1) = \{x_1, x_2, x_3\}$ *et* $X(C_2) = \{x_3, x_4, x_5\}$. *La figure 2.4 présente les trois modèles graphiques correspondants.*

2.3.2 Réseaux bayésiens

Un réseau bayésien est une représentation graphique d'un ensemble de variables aléatoires et de leurs indépendances conditionnelles sous forme d'un graphe orienté acyclique (ou *Directed Acyclic Graph* - DAG). Un réseau bayésien définit de façon complète toute distribution conjointe, à l'aide des dépendances entre les variables

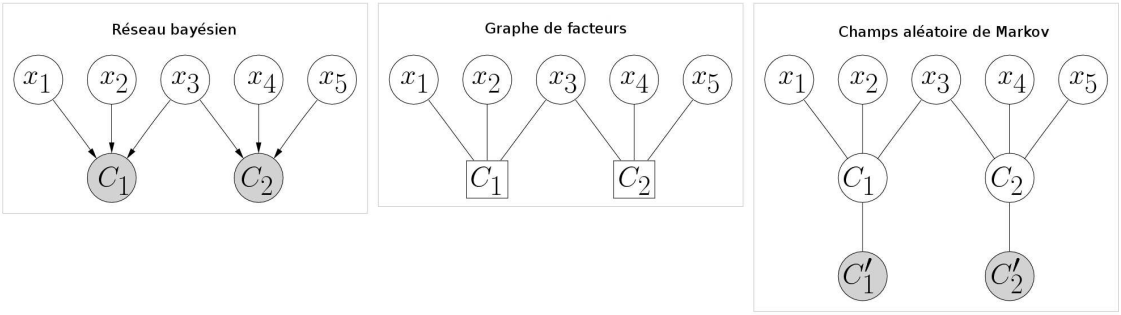


FIGURE 2.4 Modèles graphiques correspondant à l'Exemple 2

et des probabilités conditionnelles d'une variable étant données ses variables dépendantes. Dans le cas de l'exemple précédent, on remarque que le noeud de la contrainte C_1 dépend directement des variables x_1, x_2 et x_3. De plus, d'après l'algorithme de *D-Separation* [2], et selon le principe « *Explaining-Away* », on constate que les cinq variables x_1 à x_5 sont effectivement dépendantes entre elles.

Si l'on désire connaître la probabilité que la variable x_1 prenne la valeur $v \in D_1$, on peut avoir recours à une méthode d'inférence exacte (comme la méthode d'élimination de variables qui sommerait ici sur toutes les variables non observées différentes de x_1). Cependant, ce problème est en lui-même NP-difficile, et est sujet à l'explosion combinatoire avec le nombre de variables.

C'est pourquoi on fait généralement appel à des méthodes d'inférence approximées, comme la simulation de chaînes de Markov Monte-Carlo (ou *Markov Chain Monte-Carlo* - MCMC) ou la méthode *Belief-Propagation*.

2.3.3 Méthodes d'inférence approximées

Dans le cadre de la théorie probabiliste, les méthodes d'inférence approximées telles que les méthodes par passage de messages se sont révélées être particulièrement efficaces. Par transmission de messages à travers les arcs du réseau bayésien, ces méthodes permettent d'inférer l'état d'un sous-ensemble de variables en se basant sur l'ensemble des variables observées. C'est pourquoi ces méthodes sont dites des

[2] Voir le manuel *Pattern recognition and Machine Learning*, chapitre 8 (Bishop (2006)) pour plus de détails.

méthodes d'inférence.

Bien que celles-ci aient été longtemps appliquées à des problèmes d'inférence probabiliste, leur capacité à estimer les probabilités marginales font de ces méthodes des outils des plus intéressants à utiliser dans le cadre d'une recherche par retour arrière. Par exemple, la méthode *Belief-Propagation* (BP) (Pearl (1988), Kschischang *et al.* (2001)) (et par la suite *Generalized Belief-Propagation* (Yedidia *et al.* (2003))) a été originalement développée dans le but de résoudre des problèmes d'inférence probabiliste tels que ceux qui surviennent notamment en vision par ordinateur ou dans la théorie de correction d'erreur d'encodage. Kask *et al.* (2004) ont par la suite démontré que BP (aussi appelé algorithme *sum-product*) se révèle être une méthode particulièrement efficace lorsqu'appliquée à des CSPs et utilisée en tant qu'heuristique de recherche. Mezard *et al.* (2002) ont, quant à eux, inventé la méthode *Survey-Propagation* (SP), qui s'inspire de la méthode BP. SP peut être interprétée comme une généralisation de la méthode BP, pour laquelle chaque variable adopte une valeur supplémentaire, correspondant à son indifférence de se voir assigner une des autres valeurs de son domaine. Cette méthode constitue encore à l'heure actuelle rien de moins que l'état de l'art pour la résolution de problèmes de satisfaction booléenne (SAT) (Braunstein *et al.* (2005), Kroc *et al.* (2007)).

2.3.4 Méthode *Expectation-Maximization*

De son côté, la méthode d'Espérance-Maximisation (ou *Expectation-Maximization* - EM) est une méthode dite d'apprentissage statistique. L'objectif principal de cette méthode est l'apprentissage des paramètres d'un réseau bayésien. Cette méthode est un algorithme itératif permettant d'estimer les paramètres du modèle qui rendent le plus probable les variables observées (i.e. le maximum de vraisemblance), et ceci en présence de variables cachées.

Le chapitre suivant détaille l'application de cette méthode à un problème de satisfaction de contraintes.

Chapitre 3

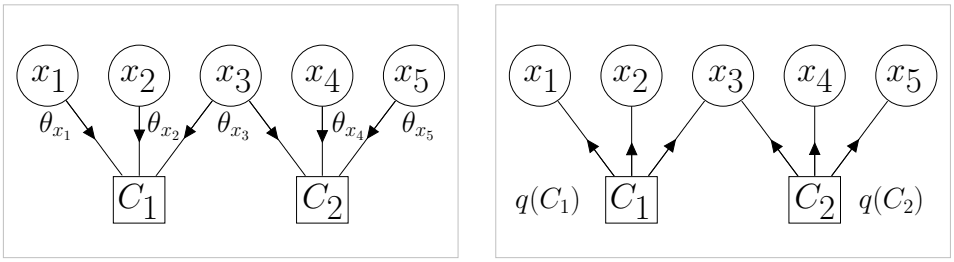
MÉTHODES EMBP

Ce chapitre introduit tout d'abord le cadre théorique des méthodes *Expectation-Maximization Belief-Propagation* (EMBP) à la section (3.1). La sous-section (3.2.1) présente les travaux antérieurs ayant proposés le cadre EMBP. Les contributions originales sont ensuite développées aux sous-sections (3.2.2) et (3.2.3), ainsi qu'à la section (3.3).

3.1 Cadre théorique

Tout comme les autres algorithmes par passage de messages, la méthode EMBP (Hsu et Mcilraith (2006), Hsu *et al.* (2007)) ajuste de façon itérative la probabilité qu'une variable se voit assigner une valeur particulière de son domaine. En reprenant la notation présentée à la section précédente, cette méthode met à jour l'ensemble des biais $\theta(x_i)$ des variables du problème en créant ainsi une étude Θ avant chaque décision de branchement. Le cadre théorique de la méthode EMBP introduit également une variable aléatoire y qui indique si chacune des contraintes est respectée. La variable y se présente donc sous la forme d'un vecteur de variables binaires, dont la taille correspond au nombre de contraintes.

La méthode EMBP procède par l'envoi de deux types de messages (ce qui fait d'elle une méthode dite par « passage de messages »), tel qu'illustré à la figure 3.1. Tout d'abord, une variable envoie sa distribution de probabilité aux contraintes dans lesquelles elle entre en jeu (Figure 3.1(a)), de telle sorte que les contraintes calculent les probabilités des tuples valides. Ensuite, les variables récupèrent cette information des contraintes associées (Figure 3.1(b)) afin de mettre à jour leur distribution de probabilité. Il est important de noter que les flèches présentes sur la figure 3.1 illustrent les messages qui sont envoyés à chaque étape, et le modèle graphique sous-jacent correspond bien à un graphe de facteur, tel que présenté au chapitre précédent.



(a) E-Step (b) M-Step

FIGURE 3.1 Illustration des messages de la méthode EMBP sur l'exemple 2

3.1.1 Formule générale de mise à jour des biais

Fondamentalement, le but principal de l'algorithme EM est de trouver l'ensemble des biais Θ qui maximise la probabilité $P(y|\Theta)$ que les contraintes soient satisfaites. Cependant, la méthodologie utilisée suppose que le vecteur de variables aléatoires y ait été originalement généré à l'aide des paramètres Θ, mais aussi à l'aide de variables cachées qui n'ont pu être observées. Dans le cadre d'un CSP, ces variables cachées (que l'on dénotera z) correspondent en fait aux configurations valides pour chacune des contraintes. On dénote par S_z le support de la distribution de z, qui correspond ainsi au produit cartésien des tuples des contraintes pour chaque solution valide du problème.

Ainsi, on désire en fait maximiser $P(y, z|\Theta)$. La difficulté majeure réside dans le fait qu'il n'est pas possible de sommer sur z puisque z n'est pas observé. À la place, on divise cette maximisation en deux phases itératives. Dans la première phase de l'algorithme EM (la phase E), on construit hypothétiquement la distribution $Q(z) = P(z|y, \Theta)$. La fonction de distribution $Q(z)$ représente la probabilité de chaque solution étant donné l'étude Θ et sachant que la variable observée y est égale à 1 (i.e. que les contraintes sont satisfaites). La figure 3.1(a) illustre cette première phase, dans laquelle chaque contrainte C reçoit la distribution de probabilité θ_{x_i} (avec $x_i \in X(C)$) et calcule la probabilité de ses configurations valides étant données ces distributions. Au cours de la deuxième phase (la phase M), l'étude Θ des variables est alors mise à jour en conséquence. Tel qu'illustré à la figure 3.1(b), les variables ajustent leur distribution en tenant compte de la probabilité des tuples valides des contraintes.

Dans le cadre de la méthode d'*Expectation-Maximization*, la phase E suppose tout

d'abord l'indépendance entre les contraintes pour calculer $Q(z)$ étant donné que l'on ne considère hypothétiquement que les configurations valides. C'est pourquoi $Q(z)$ peut s'écrire comme le produit des probabilités des contraintes, étant donnée une étude Θ. Ainsi, $Q(z) = \prod_{i=1}^{m}(q(C_i))$, avec $q(C_i)$ correspondant à la probabilité d'une configuration donnée pour la contrainte C_i.

Ensuite, la phase M applique la dépendance entre les contraintes, et fournit alors la formule générale suivante :

$$\theta_{x_i}(v) = \frac{1}{\eta} \sum_{C_k \in \mathcal{C}: x_i \in X(C_k)} \left(\sum_{z \in S_z : x_i = v} Q(z) \right) \tag{3.1}$$

Ici, η est une constante de normalisation et est égale à la sommation des valeurs du numérateur pour l'ensemble des valeurs v possibles. Hsu *et al.* (2007) fournissent davantage de détails sur la dérivation de l'équation 3.1.

Ainsi, la méthode EMBP fournit un algorithme général pour calculer une étude Θ. Dans le cadre de cette méthode, la définition de la distribution de probabilité $Q(z)$ demeure cependant non spécifiée. Les méthodes présentées dans la section suivante exploitent ce degré de liberté.

3.1.2 Application de la méthode EMBP : compromis entre précision et complexité

Une possibilité serait de déterminer explicitement S_z, l'espace des solutions du problème, pour ensuite calculer $Q(z)$ de façon exacte. Cette approche est cependant intractable puisqu'elle implique d'exprimer chaque solution du problème pour estimer les biais pour ensuite trouver une solution. Ainsi, une approximation de $Q(z)$ est requise ici.

Les méthodes décrites dans ce mémoire (regroupées dans le tableau 3.1) améliorent de façon progressive la précision de l'estimation de $Q(z)$ et, ce faisant, augmentent la complexité de calcul. À ce sujet, ces méthodes nécessitent de trouver les supports soit de façon locale (i.e. au niveau des contraintes - *Lsup*) ou de façon globale (i.e. après propagation du réseau complet de contraintes - *Gsup*) en assurant un niveau de cohérence dénoté X.

Tableau 3.1 Heuristiques de recherche EMBP

Heuristiques	Niveau de cohérence	Portée
EMBP-a	cohérence d'arc locale	contrainte `alldifferent`
EMBP-Lsup-XC	cohérence X locale	*n'importe quelle* contrainte
EMBP-Lsup-XC*	cohérence de domaine locale	contrainte `regular`
EMBP-Gsup-XC	cohérence X globale	*n'importe quel* CSP

3.1.3 Relations entre EMBP et les méthodes « variationnelles »

Comme il est souligné dans la section précédente, la méthode EMBP, bien que partant d'une méthode d'inférence exacte, se doit d'avoir recours à des hypothèses simplificatrices de façon à rester efficace. En ce sens, la méthode EMBP se rapproche des méthodes dites « variationnelles » (ou *variational methods*, Koller et Friedman (2009), Winn *et al.* (2005)) qui font partie de la famille des méthodes d'inférence approximées. De plus, typiquement, les méthodes « variationnelles » ont pour but de fournir une borne inférieure de la vraisemblance (Bernardo *et al.* (2003)), tout comme la méthode EM, qui, au lieu de maximiser directement la vraisemblance, maximise une borne inférieure de cette même vraisemblance. Jaakkola (2000) présente notamment une approche « variationnelle » qui conduit globalement à la méthode standard de l'algorithme EM, dans laquelle l'étape de maximisation (*E-Step*) est cependant adaptée.

En termes de formulation, les méthodes « variationnelles » distinguent les paramètres dits « variationnelles » des paramètres du modèle original. Cette distinction s'appuie sur le fait qu'un modèle graphique, bien que complexe en termes de densité par exemple, peut voir sa complexité sensiblement réduite lorsque le but est d'expliquer les paramètres du modèle de façon probabiliste (Jordan *et al.* (1998)). Finalement, les méthodes « variationnelles » sont définitivement omniprésentes parmi les méthodes d'inférence approximées, et même si leur fonctionnement ne reste à l'heure actuelle que partiellement expliqué, leur capacité à réduire la complexité d'un modèle graphique leur confère un potentiel très intéressant.

3.2 EMBP et cohérence locale

Tandis que l'équation (3.1) somme sur l'ensemble des probabilités de tous les tuples valides, les méthodes suivantes utilisant la cohérence locale (EMBP-a, EMBP-Lsup-XC et EMBP-Lsup-XC*) sont conçues de façon à approximer cette sommation en la simplifiant.

Afin de calculer $\theta_x(a)$ pour une contrainte donnée, le raisonnement sous-jacent à ces méthodes est de considérer toutes les autres variables x' et toutes les affectations $x' = b$ qui sont cohérentes (selon la cohérence X) avec l'affectation $x = a$ pour cette contrainte. En reprenant l'exemple 2, la figure 3.2 illustre la représentation sous forme de réseau bayésien utilisée pour estimer $Q(z)$ durant la phase E des méthodes EMBP à cohérence locale X. Ainsi, cette approche a pour but d'estimer les marginales en considérant les contraintes comme étant interdépendantes.

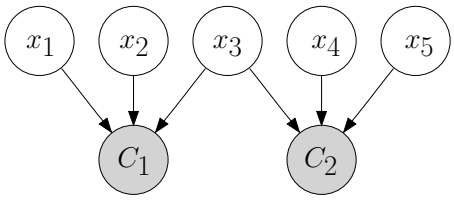


FIGURE 3.2 Réseau bayésien associé à la méthode EMBP-a pour l'exemple 2

3.2.1 EMBP-a pour la contrainte `alldifferent`

À l'origine, Hsu *et al.* (2007) suggèrent une telle approximation pour la contrainte `alldifferent`[1]. La probabilité qu'une variable x_i se voit assigner une valeur v peut être approximée par la probabilité qu'aucune autre variable de la contrainte `alldifferent` ne prenne la valeur v. Les auteurs précisent que cette approche assure ainsi une sorte de cohérence d'arc (ou *pairwise consistency*) entre x_i et les autres variables de la contrainte. En réalité, si une valeur est supprimée d'un domaine d'une variable, la réduction de domaine ne sera pas propagée : on parle alors plutôt de « *Forward-Checking* » (Haralick et Elliott (1980)) et non pas de cohérence d'arc. Ceci

[1] Cette méthode est appelée EMBP-a dans Hsu *et al.* (2007)

étant dit, en appliquant ce principe, les auteurs obtiennent la règle de mise à jour suivante pour un ensemble de contraintes `alldifferent` :

$$\begin{aligned}\theta_{x_i}(v) &= \frac{1}{\eta} \sum_{C_k \in \mathcal{C}_a : x_i \in X(C_k)} \left(\prod_{x_j \in X(C_k) \setminus x_i} \sum_{v' \in D(x_j) \setminus v} \theta_{x_j}(v') \right) \\ &= \frac{1}{\eta} \sum_{C_k \in \mathcal{C}_a : x_i \in X(C_k)} \left(\prod_{x_j \in X(C_k) \setminus x_i} \left(1 - \theta_{x_j}(v)\right) \right) \end{aligned} \tag{3.2}$$

avec η étant encore une fois la constante de normalisation.

En reprenant l'exemple 1, si l'on cherche à déterminer la probabilité que la variable x_{13} prenne la valeur 1, cela revient à supprimer la valeur 1 des domaines des variables qui interagissent avec x_{13}, tel qu'illustré à la figure 3.3.

4	3,5	1,5	1,3	2
1,3,5	3,5	1,4,5	2	1,3,4
1,3,5	2	1,4,5	1,3	1,3,4
2	1	3	4	5
1,3	4	2	5	1,3

4	3,5	1,5	1,3	2
1,3,5	3,5	1,4,5	2	1,3,4
1,3,5	2	1	1,3	1,3,4
2	1	3	4	5
1,3	4	2	5	1,3

FIGURE 3.3 Réduction de domaines associée à la méthode EMBP-a pour l'exemple 2

Ensuite, on approxime la probabilité que la variable x_{13} prenne la valeur 1 par la somme des probabilités des tuples du produit cartésien des domaines réduits pour chacune des contraintes dans lesquelles x_{13} intervient. La figure 3.4 illustre les différents termes qui constituent cette somme.

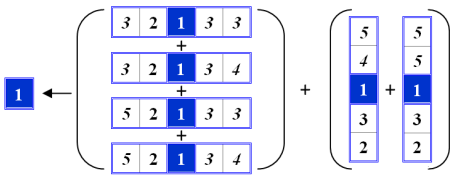


FIGURE 3.4 Termes de la somme associés à la méthode EMBP-a pour l'exemple 2

3.2.2 EMBP-Lsup-XC

Nous proposons ici de dériver des méthodes EMBP à cohérence locale X qui sont une extension relativement naturelle de EMBP-a. Le raisonnement sous-jacent à ces méthodes est le suivant : afin de calculer $\theta_x(a)$ pour une contrainte donnée, on considère toutes les affectations $x' = b$ qui sont cohérentes (selon la cohérence X) avec l'affectation $x = a$ pour cette contrainte. En d'autres termes, ces méthodes prennent en compte la réduction des domaines au niveau de la contrainte. Fondamentalement, ces méthodes procèdent une contrainte à la fois, regardent pour chaque couple variable-valeur l'ensemble des supports qui sont cohérents (selon la cohérence X) et mettent à jour les biais selon la formule suivante :

$$\theta_{x_i}(v) = \frac{1}{\eta} \sum_{C_k \in \mathcal{C} : x_i \in X(C_k)} \left(\prod_{x_j \in X(C_k) \setminus x_i} \sum_{v' \in \tilde{D}_{x_i=v}(x_j)} \theta_{x_j}(v') \right) \tag{3.3}$$

avec $\tilde{D}_{x_i=v}(x_j)$ représentant l'ensemble des valeurs restantes dans le domaine de x_j après l'affectation $x_i = v$, tel que défini à la section précédente.

En reprenant l'exemple 1, pour déterminer la probabilité que la variable x_{13} prenne la valeur 1, on filtre les valeurs qui sont incohérentes avec l'assignation $x_{13} = 1$, au sein de chacune des contraintes auxquelles x_{13} participe, tel qu'illustré à la figure 3.5.

Ensuite, on approxime la probabilité que la variable x_{13} prenne la valeur 1 par la somme des probabilités des tuples du produit cartésien des domaines réduits pour chacune des contraintes dans lesquelles x_{13} intervient. La figure 3.6 illustre les différents termes qui constituent cette somme.

Étant donné que la cohérence de paire (*pairwise consistency*) entre une variable et les autres s'apparente fortement à la cohérence d'arc, EMBP-a fait logiquement

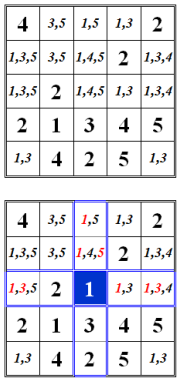


FIGURE 3.5 Réduction de domaines associée à la méthode EMBP-Lsup-XC pour l'exemple 2

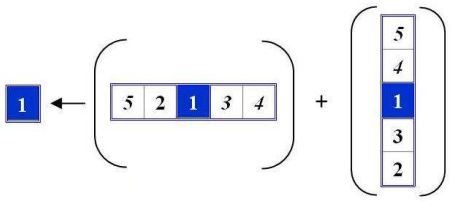


FIGURE 3.6 Termes de la somme associés à la méthode EMBP-Lsup-XC pour l'exemple 2

partie de l'ensemble des méthodes EMBP à cohérence locale X, où X représente la cohérence d'arc. Néanmoins, si l'on considère un niveau de cohérence plus fort, tel que la cohérence de domaine, l'amélioration apportée par la famille de méthodes proposée est de deux ordres :

- Tout d'abord, cela fournit une meilleure précision puisque les biais sont alors calculés à l'aide de supports qui sont cohérents selon la cohérence de domaine (et non plus selon la cohérence d'arc, comme pour EMBP-a)
- Deuxièmement, ces méthodes sont facilement implémentables pour n'importe quelle contrainte pour laquelle on connaît un algorithme capable d'assurer (efficacement) la cohérence de domaine.

De plus, cette approche peut facilement être étendue de façon à considérer n'importe quelle forme de cohérence. Tout comme la forme générale de mise à jour des

biais de la méthode EMBP, EMBP-Lsup-XC suppose toujours l'indépendance entre les contraintes au cours de la phase E, et ce n'est que la phase M qui fait respecter la dépendance entre les contraintes.

L'algorithme 1 présente le pseudo-code qui permet de mettre à jour $\theta_{x_i}(v)$ à l'itération t. P_{C_i} correspond ici à la contribution de C_i à $\theta_{x_i}(v)$ et S_{x_j} permet de stocker la somme des biais de la variable x_j. Tout d'abord, l'algorithme choisit une contrainte C_i pour laquelle l'ensemble $\mathcal{X}(x_i)$ contient x_j, puis propage l'affectation $x = v$ (ligne 3). Pour chaque variable appartenant à $\mathcal{X}(C_i)$, on somme la valeur des biais du domaine réduit (ligne 8) et on multiplie le résultat avec P_{C_i} (ligne 9). Une fois que l'algorithme a terminé de traiter la contrainte C_i, il somme la contribution de cette contrainte avec $\theta_{x_i}(v)$, puis annule l'effet de la propagation (*roll-back*) (ligne 10–11) et continue à itérer sur les contraintes qui restent à traiter. Il est à noter que lorsque tous les $\theta_{x_i}(\cdot)$ ont été calculés, ceux-ci doivent être normalisés.

```
1  θ^{t+1}_{x_i}(v) = 0;
2  pour chaque contrainte C_i ∈ C : x_i ∈ X(C_i) faire
3      assurer la cohérence X sur C_i avec x_i = v;
4      P_{C_i} = 1;
5      pour chaque variable x_j ∈ X(C_i) \ x_i faire
6          S_{x_j} = 0;
7          pour chaque valeur v' ∈ D̃_{x_i=v}(x_j) faire
8              S_{x_j} = S_{x_j} + θ^t_{x_j}(v');
9          P_{C_i} = P_{C_i} × S_{x_j};
10     θ^{t+1}_{x_i}(v) = θ^{t+1}_{x_i}(v) + P_{C_i};
11     annuler la propagation de C_i;
12 normaliser θ^{t+1}_{x_i}(v);
```

Algorithme 1 : Algorithme EMBP-Lsup-XC de mise à jour de $\theta_{x_i}(v)$ à l'itération t

Soient k le nombre de contraintes dans lesquelles x_i participe, n l'arité maximale de ces contraintes, d la cardinalité maximale des domaines des variables et P la complexité en pire cas de l'algorithme de propagation des contraintes. Alors, en pire cas, la complexité de l'algorithme 1 est $\mathcal{O}(kP + knd)$. Afin d'atteindre la convergence, le pseudo-code présenté ici doit être exécuté à chaque itération de la méthode EMBP. Pour améliorer l'efficacité de la procédure, la propagation des contraintes est réalisée une seule fois et les informations relatives à $\tilde{D}_{x_i=v}$ sont stockées en mémoire puisque

les domaines réduits demeurent constants tout au long des itérations.

Cette méthode entraîne un compromis à faire entre précision et performance. À ce sujet, il est important de noter qu'aucune propagation, mise à part celle de la contrainte en cours de traitement, n'est réalisée. D'un côté, EMBP-Lsup-XC peut possiblement fournir une information plus riche que simplement une cohérence d'arc sur les contraintes. De l'autre, elle offre une précision plus faible que si l'on assurait la cohérence de domaine au niveau du problème au complet. En termes d'efficacité cependant, propager une seule contrainte est moins coûteux que de propager le modèle complet.

3.2.3 EMBP-Lsup-XC* pour la contrainte `regular`

Tandis que la méthode précédente est une approche générale, nous proposons ici une règle de mise à jour des biais similaire, mais cette fois-ci spécifique à la contrainte `regular`, contrainte d'appartenance à un langage régulier proposée par Pesant (2004). De cette façon, on souhaite obtenir une heuristique qui, bien que non générique, soit capable de profiter davantage de la structure de la contrainte `regular` (ou *Regular Language Membership constraint*).

Soit $M = (Q, \Sigma, \delta, q_0, F)$ un automate fini déterministe (ou *Deterministic Finite Automaton* - DFA) pour lequel Q est un ensemble fini d'états, Σ est un alphabet, $\delta : Q \times \Sigma \rightarrow Q$ est une fonction de transition partielle, $q_0 \in Q$ est l'état initial, et $F \subseteq Q$ est l'ensemble des états terminaux (ou acceptants).

Étant donné un automate M, considérons le graphe en couche acyclic orienté $G = (V, A)$, construit à partir de M et pour lequel la couche $i+1$ représente l'ensemble des états de l'automate atteignable après i pas en partant de l'état initial. Selon la notation de la section précédente, on pose $X = \{x_1, x_2, ..., x_n\}$ l'ensemble des variables telles que $D_i \subseteq \Sigma$ pour $1 \leq i \leq n$. La i^{eme} variable associée à une valeur de son domaine représente alors la transition d'un état de la couche i à la couche $i + 1$. On note $v_{i,q}$ le nœud correspondant à l'état q dans la couche i et par $dest(x_1, ..., x_i)$ l'état atteint à partir de la séquence $(x_1, ..., x_{i-1})$. Pour permettre de mieux comprendre

cette notation, la figure 3.7 illustre un tel graphe. La couche L_1 correspond à l'état initial, la couche L_6 correspond à l'état final (unique ici), tandis que chaque couleur d'arête représente une valeur différente des domaines des variables. Par exemple, si on associe $\{jaune = 1, rouge = 2, vert = 3, bleu = 4\}$, alors le tuple $(1, 1, 1, 1, 2)$ associé aux variables $(x_1, x_2, x_3, x_4, x_5)$ est valide, contrairement au tuple $(1, 1, 1, 1, 3)$ qui ne satisfait pas cette contrainte.

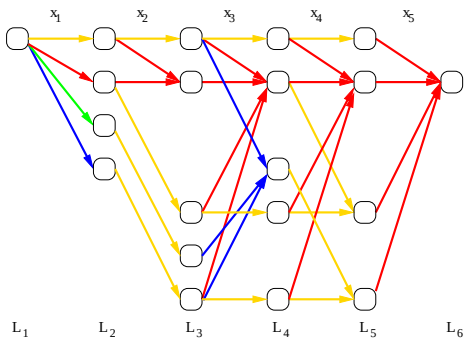


FIGURE 3.7 Exemple de graphe en couche acyclic orienté pour une contrainte `regular`

Afin d'être capable d'utiliser pleinement la structure d'un tel graphe en couche, nous introduisons également une probabilité $\theta_{x_i,q}(v)$ sur chaque arc $e : v_{i,q} \rightarrow_v v_{i+1,q'}$, correspondant à la probabilité que x_i soit égale à v et $dest(x_1, ..., x_{i-1}) = q$, et telle que :

$$\theta_{x_i}(v) = \sum_{q \in Q} \theta_{x_i,q}(v)$$

Cependant, chaque itération de l'algorithme EMBP permet de fournir une étude Θ et il n'est pas possible d'en déduire directement les biais introduits spécifiquement pour la contrainte `regular`. Ainsi, à l'itération $t + 1$, nous inférons les valeurs $\theta_{x_i,q}(\cdot)$ à partir des $\theta_{x_i}(\cdot)$ de l'itération k et selon la structure du digraphe. Ainsi, afin de distribuer la probabilité $\theta_{x_i}(v)$ parmi tous les $\theta_{x_i,q}(v)$, nous pondérons les biais des arcs selon la densité des arêtes. La densité d'une arête correspond au nombre de solutions qui utilisent cette arête sur le nombre total de solutions de cette contrainte. Dans le cadre de méthodes de dénombrement de solutions d'une contrainte, Zanarini et Pesant (2009) proposent entre autres un algorithme efficace pour déterminer cette densité spéficique à la contrainte `regular`.

Il est à noter que cette définition ne prend pas en compte les arêtes possédant plusieurs étiquettes. En se basant uniquement sur la structure du digraphe, il nous est impossible de discriminer les probabilités des deux arcs qui ont les mêmes extrémités. Étant donné que nous considérons seulement la structure de l'automate afin de déduire la mise à jour des biais des variables de la contrainte `regular`, nous supposons l'équi-répartition de la probabilité entre les arcs ayant les deux mêmes extrémités.

Sachant que l'on ne considère que les chemins qui aboutissent à un état terminal, et que chacun d'entre eux est de longueur n, la probabilité d'un arc particulier e : $v_{i,q} \rightarrow_v v_{i+1,q'}$ regroupe la probabilité de chacune des séquences $s = x_1...x_n$ telles que $dest(x_1...x_{i-1}) = q$, $x_i = v$ (i.e. les chemins qui passent par cet arc). Soit $S_{i,q,v}$ l'ensemble de ces séquences. De façon plus formelle :

$$\theta_{x_i,q}(v) = \frac{1}{\eta} \sum_{s \in S_{i,q,v}} Q(s) \tag{3.4}$$

Cependant, l'équation (3.4) requiert le calcul de chacun des tuples valides pour cette contrainte. Cette approche est clairement intractable. Afin d'être plus efficace en pratique, nous approximons l'équation ci-dessus. Au lieu de sommer sur l'ensemble des tuples valides, nous sommons sur l'ensemble des valeurs qui sont cohérentes avec celle pour laquelle nous cherchons à déterminer la probabilité. Par la suite, on notera (i, q, v) l'arc $e : v_{i,q} \rightarrow_v v_{i+1,q'}$. Soit $R_{i,q,v}(i')$ l'ensemble des arcs (i', q', v') tels qu'il existe un chemin (orienté) entre (i, q, v) et (i', q', v'). Finalement, dénotons $S^*_{i,q,v}$ l'ensemble des chemins de longueur n correspondant à $R_{i,q,v}(1) \times R_{i,q,v}(2) \times \ldots \times R_{i,q,v}(n)$.

Il est à noter que $S_{i,q,v} \subseteq S^*_{i,q,v}$. On se permet ici d'approximer l'ensemble des termes de l'équation (3.4) et cette relaxation permettra alors de simplifier la sommation.

Ainsi, nous obtenons :

$$\begin{aligned}\theta_{x_i,q}(v) &= \frac{1}{\eta} \sum_{s \in S^*_{i,q,v}} Q(s) \\ &= \frac{1}{\eta} \sum_{(1,q'_1,v'_1),\ldots,(n,q'_n,v'_n) \in S^*_{i,q,v}} \prod_{j \neq i} \theta_{x_j,q'_j}(v'_j) \\ &= \frac{1}{\eta} \prod_{j \neq i} \left(\sum_{(j,q',v') \in R_{i,q,v}(j)} \theta_{x_j,q'}(v') \right) \end{aligned} \tag{3.5}$$

Il est à noter que la factorisation est maintenant possible à l'équation (3.5) grâce à cette approximation. En effet, au lieu de considérer le sous-ensemble $S(i,q,v)$ de tuples valides, nous considérons maintenant chaque chemin s inclus dans le produit Cartésien $S^*_{i,q,v}$. De plus, pour l'affectation $x_i = v$, nous considérons chaque affectation $x_j = v'$ telle qu'il existe une séquence valide de valeurs contenant ces deux affectations. C'est pourquoi cette méthode s'apparente très fortement aux méthodes EMBP à cohérence locale précédente, puisque cette approche revient à considérer la cohérence de domaine au niveau de la contrainte `regular`. Cette méthode diffère cependant de EMBP-Lsup-XC sur deux aspects :

- Tout d'abord, nous faisons ici une différence entre $\theta_{x_i,q}(v)$ et $\theta_{x_i,q'}(v)$ selon la structure du digraphe.
- Deuxièmement, dans la phase d'initialisation, nous répartissons la probabilité $\theta_{x_i}(v)$ parmi les différents biais d'arcs $\theta_{x_i,q}(v)$ selon les densités de solutions.

Par conséquent, si, pour chaque couple variable-valeur (x_i, v), il n'y a pas plus d'un arc (i,q,v), alors cette méthode sera identique à EMBP-Lsup-XC. Cependant, dans le cas général, on peut s'attendre à une meilleure précision apportée par cette méthode, puisqu'elle capture mieux la structure de la contrainte sous-jacente.

Comme il en est fait mention dans Trick (2003), la contrainte `knapsack` présente une structure similaire au graphe en couche de la contrainte `regular`. C'est pourquoi un raisonnement analogue peut être tiré pour la contrainte `knapsack` et une règle de mise à jour des biais équivalent peut être déduite.

3.3 EMBP et cohérence globale

Introduisant une nouvelle méthode que nous appellerons EMBP-Gsup-XC, nous suggérons d'améliorer encore une fois la précision du calcul de $Q(z)$. En effet, en ce qui concerne la méthode EMBP-Gsup-XC, le problème est considéré dans son ensemble et cette approche exploite directement la dépendance entre les contraintes au moment de déterminer $Q(z)$. EMBP-Lsup-XC considère les supports qui sont cohérents pour une contrainte à la fois tandis que EMBP-Gsup-XC améliore la qualité de cette approximation en prenant en compte les supports qui sont cohérents après propagation de chaque contrainte du problème[2]. Dans la nouvelle représentation, sous-jacente à cette approximation de $Q(z)$, au lieu de considérer m contraintes différentes, nous considérons maintenant une seule contrainte, dans laquelle toutes les variables de $\mathcal{X}$ participent. La figure 3.1 illustre cette nouvelle représentation.

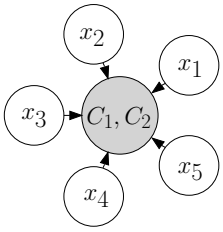


FIGURE 3.8 Réseau bayésien associé à la méthode EMBP-Gsup-XC pour l'exemple 2

La formule de mise à jour devient alors :

$$\begin{aligned} \theta_{x_i}(v) &= \frac{1}{\eta} \prod_{x_j \in \mathcal{X} \setminus x_i} \sum_{v' \in \hat{D}_{x_i=v}(x_j)} \theta_{x_j}(v') \\ &= \frac{1}{\eta} \prod_{x_j \in \mathcal{X} \setminus x_i} \left(1 - \sum_{v' \in D(x_j) \setminus \hat{D}_{x_i=v}(x_j)} \theta_{x_j}(v') \right) \end{aligned} \tag{3.6}$$

où $D(x_j)$ est le domaine de x_j avant l'affectation de $x_i = v$ et $\hat{D}_{x_i=v}(x_j)$ correspond au domaine réduit de la variable x_j après l'affectation $x_i = v$ et la propagation

[2] À noter que n'importe quel niveau de cohérence peut être considéré ici, et le niveau de cohérence peut très bien varier d'une contrainte à l'autre.

au travers de chacune des contraintes du problème.

Encore une fois, on illustre cette méthode à l'aide de l'exmple 1. Afin de déterminer la probabilité que la variable x_{13} prenne la valeur 1, on filtre les valeurs qui sont incohérentes avec l'assignation $x_{13} = 1$, mais cette fois-ci, au sein du problème au complet, tel qu'illustré à la figure 3.9.

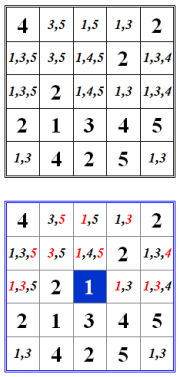


FIGURE 3.9 Réduction de domaines associée à la méthode EMBP-Gsup-XC pour l'exemple 2

Ensuite, la probabilité que la variable x_{13} prenne la valeur 1 est approximée par la somme des probabilités des tuples du produit cartésien des domaines réduits de l'ensemble des variables. La figure 3.10 illustre les différents termes qui constituent cette somme.

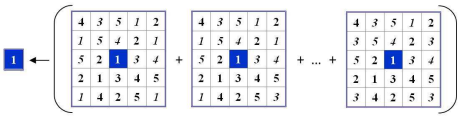


FIGURE 3.10 Termes de la somme associés à la méthode EMBP-Gsup-XC pour l'exemple 2

Cette méthode dépend indirectement de la modélisation du problème puisqu'il dépend de la puissance d'inférence qui accompagne la modélisation. Cependant, cette approche fournit également la possibilité d'utiliser différents niveaux de cohérence durant la phase de « *probing* » (pendant laquelle on calcule $\hat{D}_{x_i=v}$) et durant la phase

effective de propagation après chaque décision de branchement.

L'algorithme 2 présente le pseudo-code pour mettre à jour $\theta_{x_i}(v)$ à l'itération t. La principale différence avec l'algorithme 1 réside dans le fait qu'il n'y a plus de sommation sur l'ensemble des contraintes puisque la contribution de chaque contrainte est implicitement reflétée dans $\hat{D}_{x_i=v}(x_j)$ étant donnée la propagation initiale à l'échelle du problème au complet (ligne 2). En pratique, les expérimentations ont révélé qu'il est plus rapide d'itérer sur les éléments enlevés du domaine des variables (qui est le complémentaire de $\hat{D}_{x_i=v}(x_j)$) plutôt que sur $\hat{D}_{x_i=v}(x_j)$ lui-même.

```
1 θ^{t+1}_{x_i}(v) = 1;
2 assurer la cohérence X sur le problème complet avec x_i = v;
3 pour chaque variable x_j ∈ 𝒳 \ x_i faire
4     S_{x_j} = 1;
5     pour chaque valeur v' ∈ D(x_j) \ D̂_{x_i=v}(x_j) faire
6         S_{x_j} = S_{x_j} − θ^t_{x_j}(v');
7     θ^{t+1}_{x_i}(v) = θ^{t+1}_{x_i}(v) × S_{x_j};
8 annuler la propagation de x_i = v;
9 normaliser θ^{t+1}_{x_i}(v);
```

Algorithme 2 : Algorithme EMBP-Gsup-XC de mise à jour de $\theta_{x_i}(v)$ à l'itération t

Soient K le nombre total de contraintes, N le nombre total de variables, d la cardinalité maximale des domaines des variables et P la complexité en pire cas de l'algorithme de propagation des contraintes. Alors, en pire cas, la complexité de l'algorithme 2 est $\mathcal{O}(KP+Nd)$. Bien que cela ne soit pas flagrant lorsque l'on regarde les complexités en pire cas, l'algorithme 2 s'avère être plus gourmand en termes de temps de calcul notamment que l'algorithme 1 puisqu'habituellement $K \gg k$ et $N \gg n$. Afin d'éviter de devoir propager à chaque itération de la méthode EMBP, l'information concernant $D(x_j) \setminus \hat{D}_{x_i=v}$ est également calculée une seule fois et stockée en mémoire, comme c'était le cas pour l'algorithme 1.

Chapitre 4

RÉSULTATS EXPÉRIMENTAUX

4.1 Environnement de tests

Tous les tests ont été realisés à l'aide de Ilog Solver 6.5 et en utilisant un processeur AMD Opteron 2.4GHz avec 1GB de RAM. Pour les heuristiques qui présentent une composante aléatoire, les tests ont été exécutés dix fois sur chaque exemplaire et les moyennes arithmétiques des résultats sont alors retranscrites ici. Essentiellement, à chaque nœud de l'arbre de recherche, après avoir complété une étude à l'aide des méthodes EMBP-a, EMBP-Lsup-XC, ou EMBP-Gsup-XC, l'heuristique de recherche branche sur le couple variable-valeur qui présente le biais le plus élevé (cf. maxBias)

4.2 Description des problèmes étudiés

Cette section introduit les problèmes à partir desquels les heuristiques de recherche proposées sont évaluées.

Nonogramme Le problème du Nonogramme consiste à remplir certaines cases d'une grille rectangulaire $n \times m$ de telle sorte que les indices donnés à chaque ligne et à chaque colonne soient satisfaits. Les indices correspondent à la taille des différentes séquences de cases remplies. La figure 4.1 présente une instance sous forme graphique. Par exemple, l'indice « 1 2 » de la première colonne de l'exemple indique que sur cette colonne, il y a 2 séquences : tout d'abord (en partant du haut), on trouvera une case remplie isolée, et ensuite une séquence de deux cases remplies. Chaque séquence est séparée d'une autre séquence par au moins une case non remplie. Une fois la grille remplie, on obtient en plus un dessin monochromatique (cf. Figure 4.2, tirée de Wikipedia (2009)).

De façon plus formelle, chaque indice peut être modélisé à l'aide de la contrainte `regular`. En effet, on peut facilement exprimer l'automate correspondant qui assure

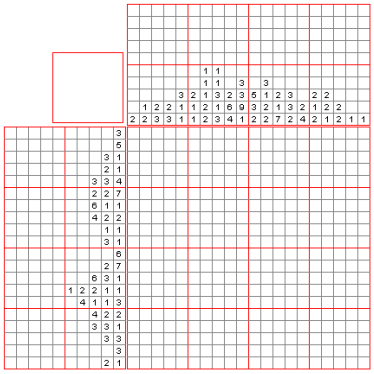


FIGURE 4.1 Exemple d'instance du problème de nonogramme

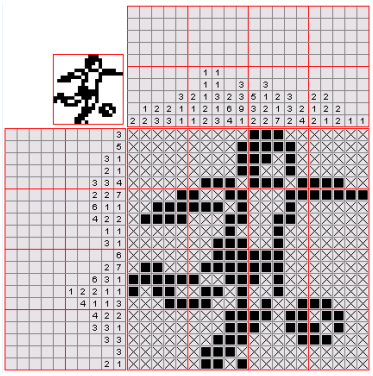


FIGURE 4.2 Exemple de solution d'une instance du problème de nonogramme

la séquence prédéfinie de remplissage des cases d'une ligne ou d'une colonne. Dans le cas de l'exemple précédent, l'automate de la figure 4.3 représente un automate possible pour l'indice « 1 2 ». Par souci de clarté, l'automate présenté ici n'est pas complet : les transitions manquantes sont alors considérées comme menant à un état non terminal.

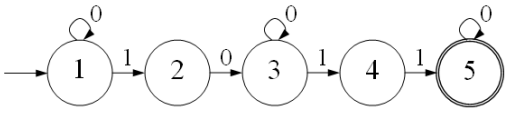


FIGURE 4.3 Automate pour la contrainte `regular` correspondant à l'indice « 1 2 »

Ainsi, le problème du Nonogramme peut se formuler de la façon suivante :

$$\begin{array}{ll} \texttt{regular}((x_{ij})_{1\leq j\leq m}, \mathcal{A}_i^r) & \forall 1 \leq i \leq n \\ \texttt{regular}((x_{ij})_{1\leq i\leq n}, \mathcal{A}_j^c) & \forall 1 \leq j \leq m \\ x_{ij} \in \{0,1\} & \forall 1 \leq i \leq n,\ 1 \leq j \leq m \end{array}$$

où $\mathcal{A}_i^r$ représente l'automate associé à la ligne i tandis que $\mathcal{A}_j^c$ représente l'automate associé à la colonne j.

Il est à noter que ces problèmes ont généralement la propriété de posséder une et une seule solution. C'est d'ailleurs le cas des instances que nous avons considérées ici.

De plus, ce problème est très homogène : toutes les contraintes sont identiques (`regular`), définies sur n ou m variables binaires, et chaque variable binaire intervient dans exactement deux contraintes.

Les expérimentations portent sur 180 instances de nonogrammes[1] de taille allant de 16×16 à 32×32. La cohérence de domaine est maintenue au niveau des contraintes et le temps limite d'exécution est fixé à dix minutes.

Carré Latin Tel que vu précédemment dans la section (2.1.2), un Carré Latin d'ordre n est défini à l'aide d'une grille $n \times n$ où les cellules d'une même ligne ou d'une même colonne prennent des valeurs différentes comprises entre 1 et n. Un quasi-groupe comportant des trous (ou Quasigroup With Holes - QWH) est un Carré Latin partiellement rempli. Le problème consiste alors à compléter un QWH de façon à obtenir un carré latin.

Parmi les applications pratiques directement reliées à ce problème figurent les tests de médicaments ou encore les exploitations agricoles. En effet, résoudre un carré latin permet notamment de savoir comment administrer n médicaments à n patients, de façon à ce que chaque patient reçoive chacun des médicaments et que deux patients ne reçoivent pas le même traitement le même jour (afin d'éviter notamment un biais lié au moment auquel le patient reçoit le médicament. Une autre application de ce problème survient dans le domaine agricole, lorsque l'exploitant désire produire n récoltes différentes tout en s'assurant que chacune des parcelles reçoive les n différentes semences au bout de n années (afin de ménager les surfaces agricoles).

[1] Instances tirées de http ://www.blindchicken.com/∼ali/games/puzzles.html

Une modélisation possible est la suivante :

$$\begin{array}{ll} \texttt{alldifferent}((x_{ij})_{1\leq j\leq n}) & \forall 1 \leq i \leq n \\ \texttt{alldifferent}((x_{ij})_{1\leq i\leq n}) & \forall 1 \leq j \leq n \\ x_{ij} = d & (i,j,d) \in S \\ x_{ij} \in \{1,2,\ldots,n\} & \forall 1 \leq i,j \leq n \end{array}$$

où S répresente l'ensemble des cellules préfixées.

Dans ce problème, toutes les contraintes sont de même type et sont définies sur n variables, sachant que chaque variable intervient dans exactement deux contraintes. Ce problème est donc également un problème très homogène.

L'ensemble des instances est le même que dans Zanarini et Pesant (2009) : 40 instances de taille $n = 30$ et ayant un pourcentage de cases non préfixées d'environ 42%, i.e. proche de la transition de phase. La transition de phase est un phénomène qui a été identifié par Cheeseman *et al.* (1991), puis caractérisé par Gomes et Selman (1997) et Gomes *et al.* (2000), et qui caractérise la difficulté de certaines instances, notamment pour le problème du carré latin. En dehors de cette transition de phase, les instances sont généralement soit aisément satisfiables ou soit facilement identifiables comme étant non satisfiables. À l'inverse, les instances qui se trouvent au niveau de la transition de phase, tels que les instances étudiées ici, se révèlent être particulièrement difficiles.

La cohérence de domaine est maintenue sur les contraintes et le temps maximal d'exécution a été fixé à 20 minutes.

Carré Magique Le problème du Carré Magique est un problème ancien, défini à l'aide d'une grille $n \times n$. Le problème consiste à placer les n^2 premiers entiers de telle façon à ce que les sommes sur les lignes, les colonnes et les diagonales soient toutes égales. Un Carré Magique partiellement rempli peut constituer un problème plus difficile à résoudre que la version traditionnelle qui débute avec une grille vide.

Deux types de contraintes interviennent ici. Ce problème est constitué de $2n + 2$ contraintes d'égalité de sac-à-dos (`equality knapsack`) régissant chacune des n variables, ainsi que d'une contrainte `alldifferent` sur l'ensemble des n^2 variables. Comparativement aux précédents problèmes, la taille des domaines des variables est substantiellement plus élevée, sachant que chaque variable peut prendre une valeur entre 1

et n^2. De façon formelle, voici le modèle proposé pour le problème du Carré Magique :

$$
\begin{array}{ll}
\texttt{alldifferent}((x_{i,j})_{1\leq i,j\leq n}) & \\
\sum_{1\leq j\leq n} x_{i,j} = sum & \forall 1 \leq i \leq n \\
\sum_{1\leq i\leq n} x_{i,j} = sum & \forall 1 \leq j \leq n \\
\sum_{1\leq i\leq n} x_{i,i} = sum & \\
\sum_{1\leq i\leq n} x_{(n+1-i),i} = sum & \\
x_{i,j} \in \{1, 2, \ldots, n^2\} & \forall 1 \leq i \leq n,\ 1 \leq j \leq n
\end{array}
$$

avec $sum = \frac{n(n^2+1)}{2}$.

Au niveau de la contrainte `alldifferent`, la cohérence de domaine est maintenue. Pour ce qui est des contraintes `knapsack` d'égalité, on maintient la cohérence de borne. L'ensemble des exemplaires qui font l'objet de tests est le même que dans Trick (2003) : 40 exemplaires ayant des cellules préfixées (dans le but d'éviter les instances triviales) — la moitié des instances ont 10 variables préfixées et l'autre moitié 50. La durée maximale d'une exécution a été fixée à une heure.

4.3 Comparaison avec d'autres heuristiques de recherche

Cette section présente les six autres heuristiques pour lesquelles nous avons souhaité comparer nos approches. Ces six heuristiques sont rndMinDom, MaxSD, IlogIBS, IlogAdvIBS, RSC-LA, and RSC2-LA.

Avant de présenter les résultats expérimentaux, voici les détails qui justifient le choix de ces six heuristiques particulières :

- rndMinDom choisit aléatoirement une variable parmi celles ayant le plus petit domaine et lui affecte une valeur quelconque appartenant à son domaine. Nous présentons les résultats obtenus à l'aide de cette heuristique afin de fournir un point de comparaison essentiel : cette heuristique, bien que clairement non informée, est largement présente dans la littérature.

- MaxSD appartient à la famille de heuristiques basées sur le dénombrement de solutions. Comme mentionné dans le chapitre 2, cette heuristique exploite l'information du dénombrement fournie par les contraintes de façon à brancher sur la partie

de l'arbre de recherche qui présente un fort potentiel de contenir un nombre élevé de solutions (Zanarini et Pesant (2009)). MaxSD est considéré comme l'état de l'art en matière de résolution de problèmes de carrés latins (QWH) difficiles.

- Les heuristiques de recherche basées sur l'impact (*Impact Based Search*, Refalo (2004)) choisissent en premier la variable dont l'assignation engendre la plus grande réduction de l'espace de recherche (le plus grand impact). Cet impact est approximé à l'aide la réduction du produit Cartésien des domaines des variables : soit par la réduction moyenne observée durant la recherche ou par le calcul exact de cette réduction. (Le calcul exact fournit effectivement une meilleure approximation mais est plus coûteux en termes de temps de calcul notamment). IlogIBS représente la méthode *Impact Based Search* pour laquelle la réduction des domaines est approximée. IlogAdvIBS choisit un sous-ensemble de cinq variables ayant le meilleur impact approximé pour ensuite choisir celle des cinq ayant le meilleur impact exact (Refalo (2004)). Les heuristiques basées sur l'impact font partie actuellement des heuristiques génériques les plus performantes.

- RSC-LA correspond à l'heuristique *Restricted Singleton Consistency Look-Ahead* (Correia et Barahona (2007)). RSC-LA maintient une cohérence singleton réduite durant la recherche tandis que RSC2-LA maintient ce niveau de cohérence pour un sous-ensemble de variables dont la taille du domaine vaut 2. Pendant que cette méthode assure la cohérence singleton, elle collecte l'information de réductions des domaines (information de *look-ahead*) sous la forme d'impacts et utilise cette information d'une manière similaire à Refalo (2004). RSC-LA s'apparente à EMBP-Gsup-XC dans le sens où ces deux méthodes réalisent une procédure de « *look-ahead* » à chaque nœud de branchement. C'est pourquoi il est essentiel de comparer les résultats de ces deux approches, puisque les méthodes ne diffèrent que par la manière dont elles agrègent l'information de « *look-ahead* ».

Enfin, lors des tests sur le problème du carré latin, nous avons ajouté l'heuristique dom/ddeg ; min conflicts qui choisit la variable possédant le plus petit ratio de la taille de son domaine sur le degré dynamique puis choisit la valeur ayant le minimum de conflits. Cette heuristique a été considérée pendant des années parmi les meilleures heuristiques spécifiques au carré latin et c'est pourquoi nous l'avons ajouté à titre de comparaison pour les résultats sur ce problème.

4.4 Résultats

Cette section présente les résultats expérimentaux obtenus sur chacun des problèmes considérés. Pour chaque heuristique, le temps CPU total moyen présenté ici correspond au temps total sur l'ensemble des exemplaires, y compris lorsque l'heuristique ne parvient pas à résoudre un exemplaire dans le temps imparti (i.e. *time-out*). Ce temps est un temps moyen sur l'ensemble des différentes exécutions pour un même exemplaire, afin de controler pour la variance des exécutions. Le nombre de retours-arrières présenté ici correspond bien à la moyenne sur toutes les exécutions et sur tous les exemplaires. Ils incorporent également les *time-outs*. Enfin, le nombre d'exemplaires résolus tient également compte des différentes exécutions pour un même exemplaire.

Nonogramme Le tableau 4.1 présente les résultats des différentes heuristiques sur les exemplaires de Nonogrammes.

TABLEAU 4.1 Temps CPU total (en secondes), nombre de *backtracks* moyen et pourcentage d'instances résolues sur 180 instances de Nonogrammes

heuristiques	temps CPU	btk	résolus
rndMinDom	20259.3	62662.4	83.8%
MaxSD	5029.6	8385	96.1%
lloglBS	741.1	2665	99.4%
llogAdvlBS	734.9	5866	99.4%
RSC-LA	580.9	10	**100.0%**
RSC2-LA	**432.5**	6	**100.0%**
EMBP-Lsup	5158.5	352	96.7%
EMBP-Lsup-XC*	7290.9	217	95.2%
EMBP-Gsup	813.9	**4**	99.4%

En ce qui concerne les exemplaires de nonogrammes, l'inférence supplémentaire fournie par les méthodes EMBP n'apporte pas d'avantage flagrant, par exemple comparativement aux méthodes traditionnelles basées sur l'impact. De plus, la spécificité de la méthode EMBP-Lsup-XC* (pour la contrainte `regular`) ne semble pas payante ici. En effet, le niveau de complexité plus élevé de cette méthode se traduit par un nombre de *backtracks* à chaque noeud de branchement plus faible mais un nombre d'exemplaires non résolus plus élevés. Il est donc difficile de tirer des conclusions

quant à la capacité de cette méthode à guider la recherche.

Le pourcentage cumulé d'instances résolues en fonction du temps apparaît à la figure 4.4. Ce graphique révèle que les exemplaires de nonogrammes sont plutôt faciles à résoudre, étant donné que la plupart d'entre elles ne requièrent que quelques secondes pour être résolues. Néanmoins, la méthode EMBP-Gsup, en dépit de son coût additionnel inhérent, performe de façon tout-à-fait comparable aux méthodes IBS et RSC-LA.

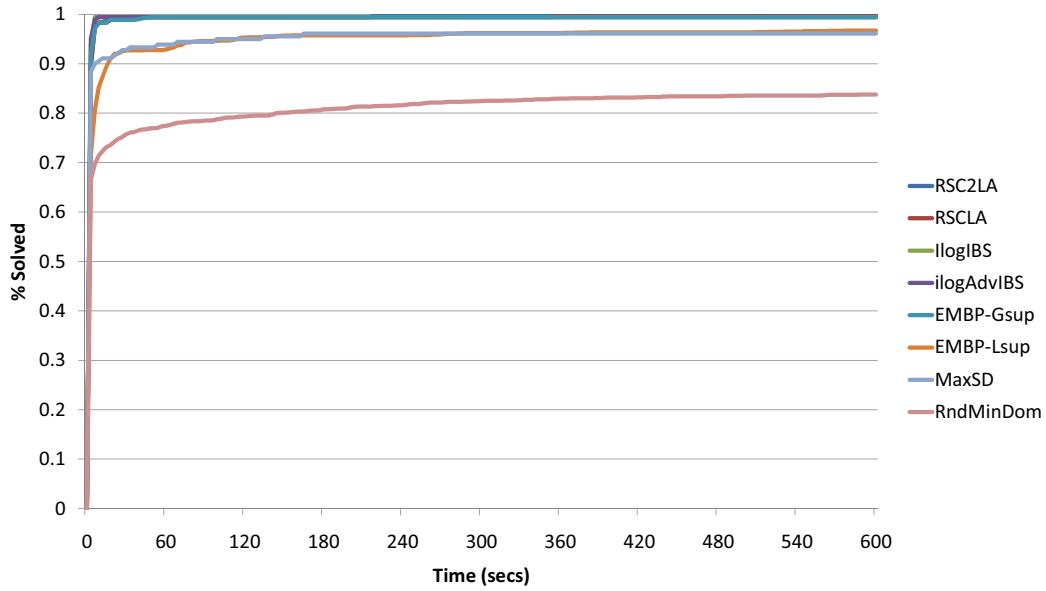


FIGURE 4.4 Pourcentage d'instances résolues en fonction du temps pour 180 nonogrammes

Au regard de ces résultats, MaxSD et EMBP-Lsup arrivent légèrement derrière, principalement dû au fait qu'elles parviennent à résoudre moins d'instances que les autres heuristiques.

Finalement, la méthode rndMinDom qui constitue notre point de repère arrive bon dernier pour chacun des critéres considérés. Enfin, un élément important qui apparaît dans le tableau mais pas sur la figure est la différence dans le nombre de *backtracks* pour EMBP-Gsup par rapport aux autres méthodes. En effet, les résultats du tableau soulignent que trois ordres de magnitude séparent EMBP-Gsup des autres heuristiques, à l'exception des deux basées sur RSC-LA. Néanmois, cela ne se traduit pas par le meilleur temps total, à cause d'une instance qui n'a pas pû être résolue dans le temps

accordé.

À titre indicatif, le tableau 4.2 présente la moyenne des écart-types sur les dix exécutions pour chacune des 180 instances de Nonogrammes. Ce tableau souligne que, bien que non déterministes, ces méthodes présentent toutefois une variance relativement faible, au regard de leurs résultats respectifs.

TABLEAU 4.2 Écart-type moyen sur l'ensemble des exécutions pour le temps CPU total (en secondes) et le nombre de *backtracks* moyen sur les instances de Nonogrammes

heuristiques	temps CPU	btk
EMBP-Lsup	6.8	51.1
EMBP-Lsup-XC*	2.6	43.9
EMBP-Gsup	0.1	1.2

Carré Latin Le tableau 4.3 présente le temps CPU total, le nombre de *backtracks* moyen et le pourcentage d'instances résolues, obtenus par les différentes heuristiques sur les instances de Carré Latin.

TABLEAU 4.3 Temps CPU total (en secondes), nombre de *backtracks* moyen et pourcentage d'instances résolues sur 40 instances difficiles d'ordre 30 de Carrés Latins

heuristiques	temps CPU	btk	résolus
rndMinDom	26328.2	1300056	56.8%
MaxSD	4939.1	3503	**100.0%**
llogIBS	29017.0	1001570	45.0%
llogAdvIBS	13795.8	914849	85.0%
RSC-LA	14019.9	856	95.0%
RSC2-LA	7178.7	4880	95.0%
EMBP-Lsup-DC	12814.2	6642	82.5%
EMBP-Gsup-DC	**3946.9**	**55**	99.5%
EMBP-a	13932.2	82158	79.0%
dom/ddeg ; min conflicts	19070.7	788887	75.0%

À la lumière de ces résultats, on constate que EMBP-Gsup réalise la meilleure performance en termes de *backtracks*, présentant un nombre de *backtracks* moyen qui est au moins un ordre de magnitude inférieur à chacune des autres heuristiques considérées. De plus, EMBP-Gsup réalise le meilleur temps total sur l'ensemble des

exemplaires. Encore une fois, la majeure partie du temps de calcul est consacré à la propagation du problème au complet à chaque noeud de branchement de l'arbre de recherche. Cependant, on constate que la qualité des biais produits par cette méthode est définitivement payante.

Le coût additionnel induit par EMBP-Gsup est facilement observable à la figure 4.5. L'heuristique requiert en moyenne 50 secondes avant même d'atteindre une feuille de l'arbre de recherche. Par conséquent, l'heuristique ne résout quasiment aucune instance en deçà d'une minute. Par la suite, elle est capable toutefois de résoudre environ 75% des exemplaires en 90 secondes.

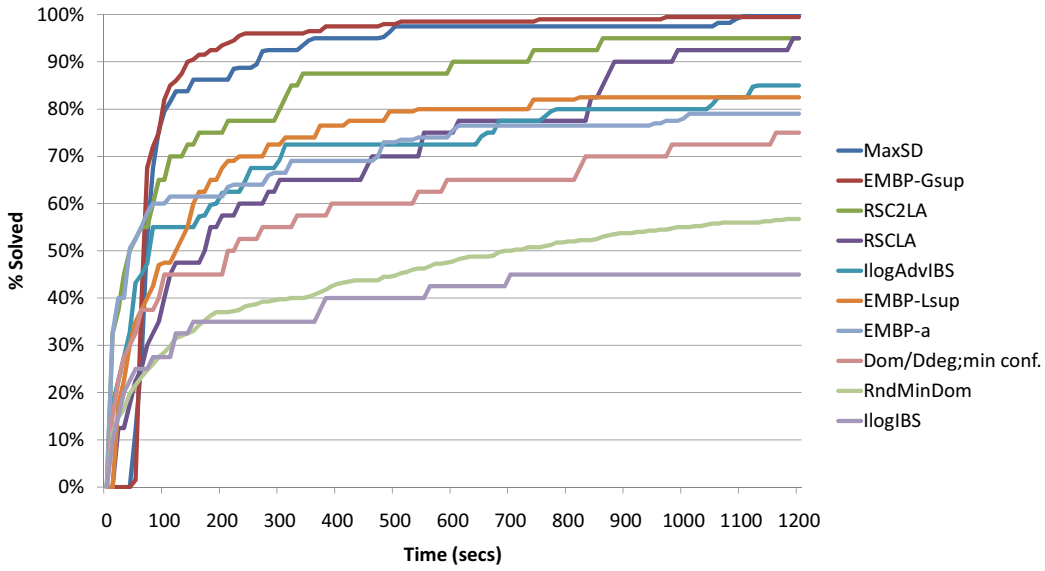


FIGURE 4.5 Pourcentage d'instances résolues en fonction du temps pour 40 carrés latins d'ordre 30

MaxSD présente un comportement similaire, notamment dû à l'algorithme d'échantillonnage utilisé pour compter le nombre de solutions des contraintes `alldifferent`. Néanmoins, le temps total requis par EMBP-Gsup-DC pour venir à bout des 40 exemplaires est significativement plus faible que pour MaxSD. Les heuristiques qui se sont révélées avoir le meilleur temps de calcul pour les instances de nonogrammes (i.e. les heuristiques basées sur RSC-LA et IBS) s'avèrent avoir plus de difficulté ici, avec un temps total allant de deux (2) à sept (7) fois plus important que EMBP-Gsup, pour finalement résoudre le même nombre d'instances. Finalement, il est intéressant de comparer l'approche proposée par Hsu *et al.* (2007) par rapport à EMBP-Lsup.

Les deux méthodes sont similaires (même si la première ne s'applique qu'à la contrainte `alldifferent`), et la seule différence vient du fait que EMBP-Lsup maintient une cohérence de domaine sur les contraintes tandis que EMBP-a assurent une forme faible de cohérence d'arc. Selon les résultats obtenus sur les instances du Carré Latin, EMBP-a est capable de réaliser davantage de retours-arrières comparativement à EMBP-Lsup pour des temps totaux similaires, mais celle-ci semble prendre des décisions de branchement moins informées que EMBP-a.

Le tableau 4.4 présente la moyenne des écart-types sur les dix exécutions pour chacune des 40 instances de Carrés Latins. On observe que la variance des résultats reste faible par rapport aux résultats moyens, bien qu'elle soit plus élevée que pour les problèmes de Nonogrammes. Toutefois, cela ne remets pas en question ici la dominance de la méthode EMBP-Gsup-DC sur les méthodes autres que MaxSD.

TABLEAU 4.4 Écart-type moyen sur l'ensemble des exécutions pour le temps CPU total (en secondes) et le nombre de *backtracks* moyen sur les instances de Carrés Latins

heuristiques	temps CPU	btk
EMBP-Lsup-DC	6.1	483.8
EMBP-Gsup-DC	58.1	72.2
EMBP-a	17.0	2080.1

Carré Magique Le tableau 4.5 présente les résultats des différentes heuristiques sur les exemplaires de Carrés Magiques.

De la même façon que pour les deux précédents problèmes, la figure 4.6 présente le pourcentage cumulé du nombre d'exemplaires résolus en fonction du temps d'exécution.

EMBP-Gsup se révèle être l'heuristique la plus performante sur l'ensemble des instances de Carré Magique considérées, avec un temps total inférieur de plus de 35% sur la deuxième heuristique (ilogAdvIBS). Comme le montre la figure 4.6, EMBP-Gsup est efficace à la fois sur les instances difficiles et sur les instances faciles. Encore une fois, EMBP-Gsup présente le plus faible nombre de *backtracks* parmi l'ensemble des heuristiques, avec une différence d'au moins un ordre de magnitude.

TABLEAU 4.5 Temps CPU total (en secondes), nombre de *backtracks* moyen et pourcentage d'instances résolues sur 40 Carrés Magiques

heuristiques	temps CPU	btk	résolus
rndMinDom	7397.0	4018251	97.0%
MaxSD	8895.7	242290	95.0%
IlogIBS	72078.2	22396381	50.0%
IlogAdvIBS	5067.9	2224191	97.5%
RSC-LA	39612.4	48759	75.0%
RSC2-LA	34524.3	1180456	80.0%
EMBP-Lsup	98910.7	20572	43.0%
EMBP-Gsup	**3758.2**	**895**	**98.8%**

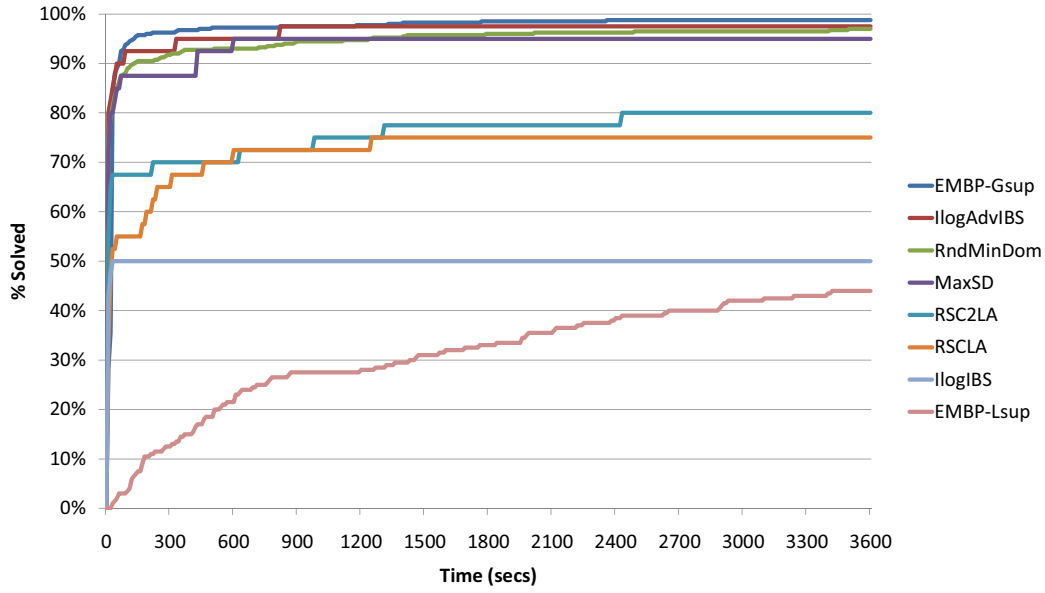


FIGURE 4.6 Pourcentage d'instances résolues en fonction du temps pour 40 Carrés Magiques

Même si elle obtient de bons résultats sur les instances du Carré Latin, MaxSD requiert cette fois si le double de temps sans atteindre le même nombre d'instances résolues que EMBP-Gsup.

Comme précédemment, EMBP-Lsup ne parvient pas à une performance intéressante par rapport à EMBP-Gsup. Cependant, il est important de rappeler ici que EMBP-Gsup bénéficie d'une forme de cohérence singleton qui n'est pas maintenue par EMBP-Lsup.

Le tableau 4.6 présente la moyenne des écart-types sur les dix exécutions pour chacune des 40 instances de Carrés Magiques. La variance relativement élevée des résultats de la méthode EMBP-Gsup s'explique en partie par les quelques exécutions qui ont échouées. Cependant, cela ne vient pas contester la supériorité de la méthode EMBP-Gsup-DC pour le problème du Carré Magique.

TABLEAU 4.6 Écart-type moyen sur l'ensemble des exécutions pour le temps CPU total (en secondes) et le nombre de *backtracks* moyen sur les instances de Carrés Magiques

heuristiques	temps CPU	btk
EMBP-Lsup-DC	1174.7	10683.9
EMBP-Gsup-DC	103.3	1213.6

Finalement, EMBP-Gsup constitue l'heuristique de recherche la plus consistante sur l'ensemble des problèmes considérés, et ce, comparativement à ce qui constitue (selon le meilleur de nos connaissances) à l'état de l'art en matière d'heuristiques pour ces mêmes problèmes. RSC-LA obtient des résultats légèrement meilleurs sur les nonogrammes, mais pas sur les carrés latins, et ne parvient pas à résoudre 20% des instances de Carré Magique. Les heuristiques basées sur IBS réalisent une performance comparable à EMBP-Gsup sur les nonogrammes, mais arrivent loin derrière sur les carrés magiques. Enfin, MaxSD obtient de bons résultats sur les carrés latins mais ne fait pas aussi bien que EMBP-Gsup sur les deux autres problèmes considérés.

Chapitre 5

CONCLUSION

5.1 Discussion

Dans cette section, nous établissons des liens entre certaines heuristiques qui sont utilisées dans le cadre de ce mémoire.

Les algorithmes de dénombrement de solutions (ou *Solution-Counting algorithms*) (Zanarini et Pesant (2009)) proposent une approche qui s'apparente à la méthode EMBP. En effet, l'algorithme de dénombrement de solutions basé sur les contraintes fournit également les marginales des variables pour une contrainte spécifique. Cette méthode considère cependant les contraintes séparément. Dans le cadre théorique de la méthode EMBP, cela revient à considérer un graphe de facteurs indépendant pour chacune des contraintes, tel que présenté à la figure 5.1 qui reprend l'exemple 2.

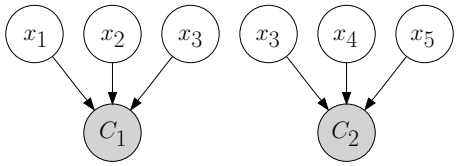


FIGURE 5.1 Réseau bayésien associé aux méthodes de dénombrement de solutions pour l'exemple 2

Ainsi, l'algorithme de dénombrement de solutions basé sur les contraintes estime les distributions marginales de contraintes *a priori* indépendantes. Au sein d'un arbre de recherche par retour arrière, cette approche suggère alors de considérer une agrégation simple de ces marginales, mais omet cependant un raisonnement plus général qui prend en compte l'indépendance entre les contraintes.

En outre, la première itération de la méthode EMBP à cohérence locale X parviendrait au calcul exact des densités de solutions telles que définies dans Zanarini et Pesant (2009) si nous étions en mesure de calculer $Q(z)$ avec exactitude (à condition toute-

fois que l'on pose à nouveau l'hypothèse que les biais des variables sont initialisées uniformément).

Les méthodes `RSC-LA` proposées par Correia et Barahona (2007) présentent également beaucoup de similarités avec EMBP-Gsup-XC. Dans le cadre de ces travaux, nous nous intéressons à la construction d'heuristiques de recherche utilisant des informations de *look-ahead* qui fournissent en retour une cohérence singleton restreinte (ou *restricted singleton consistency*). À l'inverse, dans Correia et Barahona (2007), les auteurs maintiennent le niveau de cohérence singleton restreint et se servent de l'information fournie lors des procédures de *look-ahead* pour dériver des heuristiques de recherche. Si l'on se compare à Correia et Barahona (2007), lorsque nous calculons EMBP-Gsup-XC, nous bénéficions également de la cohérence singleton restreinte qui nous permet d'élaguer l'arbre de recherche à l'aide de chaque paire variable-valeur qui est décelée comme étant incohérente. Ainsi, les deux méthodes requièrent un coût additionnel (ou *overhead*) en terme de temps de calcul à chaque nœud. La différence entre les heuristiques de recherche réside dans le fait que Correia et Barahona (2007) exploitent l'information des impacts tandis que nous nous appuyons sur le raisonnement par inférence.

Par ailleurs, il existe des parallèles intéressants entre `IBS` et les méthodes EMBP. Dans le cas où nous serions amenés à initialiser uniformément les biais des variables, la première itération de EMBP-Gsup-XC calculerait l'impact de chacune des paires variable-valeur comme étant la réduction du produit cartésien du domaine, à la manière d'IBS (Refalo (2004)) ou de RSC-LA (Correia et Barahona (2007)). Par la suite, les itérations subséquentes raffinent cet impact, générant ainsi ce que l'on peut considérer comme étant un « impact pondéré ».

5.2 Conclusion et pistes de recherche

Ces travaux ont permis de fournir des heuristiques de recherche génériques et efficaces construites à partir du cadre méthodologique proposé par la méthode EMBP. De plus, les méthodes EMBP précédemment proposées par Hsu *et al.* (2007) s'intéressaient exclusivement à des problèmes n'impliquant que des contraintes `alldifferent`.

Une des contributions de ce mémoire est de s'affranchir de cette restriction. En outre, nous avons proposé une formulation qui s'avère davantage efficace et qui présente une performance prometteuse, comparable à celle de l'état de l'art (elle s'est révélée être la plus robuste sur l'ensemble des problèmes traités dans ces travaux). Les heuristiques de recherches proposées ici ont également pour avantage de générer systématiquement moins de retours arrières. Ceci indique d'une part que les heuristiques guident bien la recherche et d'autre part qu'il y a de la place pour une potentielle amélioration en termes de temps de calcul, notamment si certains calculs sont optimisés ou approximés (voir paragraphe suivant). Une avancée importante a été réalisée lorsque l'on compare aux approches présentées dans Hsu *et al.* (2007) et Zanarini et Pesant (2009). En effet, tandis que ces deux précédentes approches requièrent respectivement des règles de mise à jour spécifiques aux contraintes et des algorithmes de dénombrement de solutions basés sur des contraintes particulières, EMBP-Gsup-XC et EMBP-Lsup-XC sont totalement générales et facilement utilisables pour n'importe quel modèle.

Il convient de noter à ce stade que d'importantes questions demeurent ouvertes. Tout d'abord, même si des biais précis apportent assurément de l'information utile pour une heuristique de recherche, la meilleure façon de l'utiliser reste à déterminer. En effet, lorsqu'elle est utilisée dans le cadre d'une heuristique de choix de variable, une méthode générant une étude Θ complète nécessite tout de même que l'on définisse une stratégie de branchement. Les possibilités sont alors nombreuses. Nous pourrions par exemple choisir de brancher sur le biais le plus élevé, comme nous l'avons fait lors de ces expérimentations. Nous aurions cependant tout aussi bien pu choisir de brancher sur le biais ayant la plus grande force (ou *strength*, i.e. la différence entre un biais et l'inverse de la taille du domaine), ou bien encore exclure du domaine d'une variable la valeur ayant le biais le plus faible.

Ensuite, au lieu de calculer une étude à chaque nœud de l'arbre de recherche, nous pourrions utiliser de l'information calculée antérieurement. À titre d'exemple, nous pourrions établir plus d'une variable à la fois, ou encore calculer des études partielles et sauver l'information de réduction de domaine, à la manière d'une heuristique `IBS`.

Dans le cadre de travaux futurs, il serait intéressant d'observer/quantifier la corrélation entre la qualité des estimations des distributions des variables par rapport aux marginales exactes et la qualité de l'heuristique. Une corrélation positive a été

identifiée lors de travaux menés par Hsu *et al.* (2008) sur des problèmes SAT. Nous aimerions étendre ces résultats pour des distributions de variables non nécessairement booléennes. Au niveau théorique, on exprime généralement la distance entre deux distributions de probabilités discrètes à l'aide de la divergence de *Kullback-Leibler* (Kullback et Leibler (1951)). Pour des variables à domaine de taille quelconque, on peut alors utiliser ici cette divergence selon deux approches. La première revient à considérer la distribution d'une variable sur son support au complet. La seconde prend en considération le choix de l'heuristique, en redéfinissant la distribution de cette variable selon une loi de Bernoulli, pour laquelle la probabilité de succés correspond à la probabilité de la valeur choisie par l'heuristique. L'étude de cette corrélation pourrait notamment apporter des éléments de réponse à la question posée précédemment, qui était de savoir comment utiliser de manière adéquate ces estimations.

Une autre piste qui semble prometteuse serait de calculer différentes *études*, notamment en haut de l'arbre de recherche. La principale motivation réside dans le fait qu'une *étude* représente une certaine configuration des variables, de telle sorte que cette configuration corresponde vraisemblablement à une solution du CSP. En aggrégeant les résultats de plusieurs études, on tend alors à se protéger d'une situation où l'étude favorise une configuration pour laquelle quasiment toutes les contraintes sont entièrement satisfaites (la vraisemblance est élevée) mais qui ne posséde aucune solution (une contrainte « refuse » d'être satisfaite). Cette allée de recherche est étroitement liée avec l'identification de la structure sous-jacente du problème, et semble également prometteuse afin de déceler certaines caractéristiques du problème ou des variables du problème (comme par exemple le caractère *backbone* de celles-ci).

Enfin, nous souhaiterions combiner les méthodes EMBP avec les méthodes de dénombrement de solutions, afin de profiter à la fois du raisonnement probabiliste et de l'aggrégation de l'information de plusieurs contraintes des méthodes EMBP et de la qualité des approximations de l'espace des solutions fournies par les méthodes de dénombrement. Lors de travaux futurs, nous aimerions également dériver des règles de mise à jour équivalentes pour le EMSP (*Expectation-Maximization Survey Propagation* Hsu *et al.* (2008) et exploiter le potentiel prometteur du cadre proposé par la méthode *Survey Propagation*.

Références

BERNARDO, J. M., BAYARRI, M. J., BERGER, J. O., DAWID, A. P., HECKERMAN, D., SMITH, A. F. M., (EDS, M. W., BEAL, M. J. et GHAHRAMANI, Z. (2003). The variational bayesian em algorithm for incomplete data : with application to scoring graphical model structures.

BESSIÈRE, C. et RÉGIN, J.-C. (1996). Mac and combined heuristics : Two reasons to forsake fc (and cbj ?) on hard problems. E. C. Freuder, éditeur, *CP*. Springer, vol. 1118 de *Lecture Notes in Computer Science*, 61–75.

BISHOP, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

BOUSSEMART, F., HEMERY, F., LECOUTRE, C. et SAIS, L. (2004). Boosting systematic search by weighting constraints. R. L. de Mántaras et L. Saitta, éditeurs, *ECAI*. IOS Press, 146–150.

BRAUNSTEIN, A., MÉZARD, M. et ZECCHINA, R. (2005). Survey propagation : An algorithm for satisfiability. *Random Structures Algorithms*, 27, 201–226.

CAMBAZARD, H. et JUSSIEN, N. (2005). Identifying and exploiting problem structures using explanation-based constraint programming. R. Barták et M. Milano, éditeurs, *CPAIOR*. Springer, vol. 3524 de *Lecture Notes in Computer Science*, 94–109.

CHEESEMAN, P., KANEFSKY, B. et TAYLOR, W. M. (1991). Where the really hard problems are. *IJCAI'91 : Proceedings of the 12th international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 331–337.

CORREIA, M. et BARAHONA, P. (2007). On the integration of singleton consistencies and look-ahead heuristics. *CSCLP*. 62–75.

FREUDER, E. (1996). In pursuit of the holy grail. *ACM Comput. Surv.*, 63.

FROST, D. et DECHTER, R. (1995). Look-ahead value ordering for constraint satisfaction problems. *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'95*. Montreal, Canada, 572–578.

GINSBERG, M., FRANK, M., HALPIN, M. et TORRANCE, M. (1990). Search lessons learned from crossword puzzles. *In Proceedings of the Eighth National Conference on Artificial Intelligence*. 210–215.

GOLOMB, S. W. et BAUMERT, L. D. (1965). Backtrack programming. *J. ACM*, 12, 516–524.

GOMES, C. P. et SELMAN, B. (1997). Problem structure in the presence of perturbations. *AAAI/IAAI*. 221–226.

GOMES, C. P., SELMAN, B., CRATO, N. et KAUTZ, H. A. (2000). Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *J. Autom. Reasoning*, 24, 67–100.

HARALICK, R. M. et ELLIOTT, G. L. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artif. Intell.*, 14, 263–313.

HSU, E. I., KITCHING, M., BACCHUS, F. et MCILRAITH, S. A. (2007). Using expectation maximization to find likely assignments for solving csp's. *AAAI*. 224–230.

HSU, E. I. et MCILRAITH, S. A. (2006). Characterizing propagation methods for boolean satisfiability. *Proc. of 9th International Conference on Theory and Applications of Satisfiability Testing (SAT'06*. 325–338.

HSU, E. I., MUISE, C. J., BECK, J. C. et MCILRAITH, S. A. (2008). Applying probabilistic inference to heuristic search by estimating variable bias. *Proceedings of the 1st International Symposium on Search Techniques in Artificial Intelligence and Robotics (at AAAI08)*. Chicago, IL, USA, vol. WS-08-10, 68 – 75.

JAAKKOLA, T. S. (2000). Tutorial on variational approximation methods. *In Advanced Mean Field Methods : Theory and Practice*. MIT Press, 129–159.

JORDAN, M. I., GHAHRAMANI, Z., JAAKKOLA, T. S. et SAUL, L. K. (1998). An introduction to variational methods for graphical models. *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*. Kluwer Academic Publishers, Norwell, MA, USA, 105–161.

KASK, K., DECHTER, R. et GOGATE, V. (2004). Counting-based look-ahead schemes for constraint satisfaction. *Proc. of 10th International Conference on Constraint Programming (CP'2004)*. 317–331.

KILBY, P., SLANEY, J. K., THIBAUX, S. et WALSH, T. (2005). Backbones and backdoors in satisfiability. M. M. Veloso et S. Kambhampati, éditeurs, *AAAI*. AAAI Press / The MIT Press, 1368–1373.

KOLLER, D. et FRIEDMAN, N. (2009). *Probabilistic Graphical Models : Principles and Techniques*. MIT Press.

KROC, L., SABHARWAL, A. et SELMAN, B. (2007). Survey propagation revisited. *23rd UAI*. Vancouver, BC, 217–226.

KSCHISCHANG, F. R., FREY, B. J. et LOELIGER, H. A. (2001). Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47, 498–519.

KULLBACK, S. et LEIBLER, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22, 79–86.

LEBRAS, R., ZANARINI, A. et PESANT, G. (2009). Efficient generic search heuristics within the embp framework. I. P. Gent, éditeur, *CP*. Springer, vol. 5732 de *Lecture Notes in Computer Science*, 539–553.

MEZARD, M., PARISI, G. et ZECCHINA, R. (2002). Analytic and algorithmic solution of random satisfiability problems. *Science*, 297, 812–815.

MONASSON, R., ZECCHINA, R., KIRKPATRICK, S., SELMAN, B. et TROYANSKY, L. (1999). 2+p-sat : Relation of typical-case complexity to the nature of the phase transition. *Random Struct. Algorithms*, 15, 414–435.

PARKES, A. J. (1997). Clustering at the phase transition. *In Proc. of the 14th Nat. Conf. on AI*. AAAI Press / The MIT Press, 340–345.

PEARL, J. (1988). *Probabilistic Reasoning in Intelligent Systems :Networks of Plausible Inference*. Morgan Kaufmann.

PESANT, G. (2004). A regular language membership constraint for finite sequences of variables. M. Wallace, éditeur, *CP*. Springer, vol. 3258 de *Lecture Notes in Computer Science*, 482–495.

PESANT, G. et QUIMPER, C.-G. (2008). Counting solutions of knapsack constraints. L. Perron et M. A. Trick, éditeurs, *CPAIOR*. Springer, vol. 5015 de *Lecture Notes in Computer Science*, 203–217.

REFALO, P. (2004). Impact-based search strategies for constraint programming. *CP*. 557–571.

ROSSI, F., BEEK, P. V. et WALSH, T. (2006). *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA.

TRICK, M. A. (2003). A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals of Operations Research*. vol. 118, 73–84.

WIKIPEDIA (2009). Nonogram — wikipedia, the free encyclopedia. [Online ; accessed 22-June-2009].

WINN, J., BISHOP, C. M. et JAAKKOLA, T. (2005). Variational message passing. *Journal of Machine Learning Research*, 6, 661–694.

YEDIDIA, J. S., FREEMAN, W. T. et WEISS, Y. (2003). *Understanding belief propagation and its generalizations*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

ZANARINI, A. et PESANT, G. (2007). Solution counting algorithms for constraint-centered search heuristics. *CP*. 743–757.

ZANARINI, A. et PESANT, G. (2009). Solution counting algorithms for constraint-centered search heuristics. *Constraints*, 14, 392–413.